PROYECTO FIN DE GRADO

Universidad Carlos III de Madrid

ESCUELA POLITÉCNICA SUPERIOR INGENIERÍA EN INFORMÁTICA

GESTIÓN MULTI-VARIABLE INTELIGENTE DE EQUIPOS MULTI-DISCIPLINARIOS E INTERNACIONALES DE EQUIPOS Y PROYECTOS DE DESARROLLO DE SOFTWARE

Realizado por:
Ana Luque Calvo

Supervisado por:
D. Juan Miguel Gómez Berbis

Febrero, 2013

Agradecimientos

Este proyecto no habría sido posible sin el apoyo de mis padres y mis hermanos. Gracias a ellos soy lo que soy y he tenido la oportunidad de realizar este proyecto. Sin ellos, esto no hubiera sido posible. Gracias.

Por supuesto, gracias a todos mis compañeros de clase, por ayudarme, acompañarme y haberme hecho pasar inmejorables momentos durante estos años. Raúl, Jorge, Félix, Toni, Machi, Javi, Lonchez, Agui, Antonio, Elena, Quique, Watto, muchísimas gracias.

Mención especial para Francisco Sánchez Navarro, compañero de colegio y de universidad. Mi compañero de prácticas a lo largo de la carrera, mi compañero de sufrimientos y alegrías. Mil gracias.

Por supuesto, no podían faltar todas mis amigas que siempre han estado dándome su apoyo y ánimos: Ana, Pili, Raquel, Patri, Lore, Zaira, Paula y Silvia, y todos mis amigos del baloncesto, sobre todo Raúl, Juanpe, Juanse y Pedro.

No puedo olvidarme de agradecer a mi tutor Juan Miguel Gómez Berbis, el haberme dado la oportunidad de realizar el proyecto con él, por haber aportado la idea, sugerencias, comentarios y sobre todo por su paciencia.

Por último, recordar a todos los que de alguna manera han ayudado a la realización de este proyecto, profesores, compañeros, personal administrativo, personal de la cafetería, todos en cierta manera habéis aportado a este proyecto.

Muchas gracias a todos

Resumen

Este proyecto abarca por un lado el campo de ingeniería del software, y por otro lado, el campo de la Inteligencia Artificial (IA) dentro de la rama de aprendizaje automático.

El objetivo principal de este proyecto ha sido desarrollar un sistema que sea capaz de predecir y calcular la productividad generada por un equipo de trabajo a la hora de desarrollar un proyecto antes de formar el equipo de trabajo y comenzar el desarrollo del proyecto. Además, se predecirá la formación idónea de cada equipo de trabajo en función del proyecto dado.

Nuestro sistema lo hemos basado en un algoritmo genético simple cuya función fitness ha sido calculada a partir de una red de neuronas del tipo perceptrón multicapa. La función principal de la red de neuronas es analizar la relación que existe entre las características de los miembros del equipo. Por otro lado, el algoritmo genético se encargará de seleccionar los atributos más válidos de los individuos.

El proceso de experimentación demuestra que la solución generada por el algoritmo genético es capaz de proporcionar, ante nuevos patrones de datos y bajo el mínimo error posible, el resultado esperado por el usuario.

Palabras clave: ingeniería del software, Inteligencia artificial, metodología desarrollo, aprendizaje automático, red neuronal artificial, algoritmos genéticos, clasificación de datos.

Abstract

This project covers on one hand the software engineering field, and on the other hand, the field of Artificial Intelligence (AI) within the branch of automatic learning.

The main objective of this project is developing a system that is able to predict the productivity generated by a team when developing a project before forming the team and start the project. Moreover the productivity of each team, the system will combine the employees in order to find the best team composition for a given project.

Our system has been based on a simple genetic algorithm whose fitness function has been derived from a neural network based on a multilayer perceptron. The main function of the neural network is to analyze the relationship between the characteristics of team members. Furthermore, the genetic algorithm will select the best attributes of individuals.

The process of experimentation demonstrated that the solution generated by the genetic algorithm is capable of providing to new input patterns and under the least possible error, the result expected by the user.

Keywords: software engineering, artificial intelligence, development methodology, automatic learning, artificial neural networks, genetic algorithms, data classification.

Contenido

Índice de Tablas

Índice de Figuras

PARTE I: INTRODUCCIÓN

Capítulo 1 | Introducción

El proyecto de *Gestión multi-variable de equipos multi-disciplinarios e internacionales de equipos y proyectos de desarrollo de software* nace con la convicción de que el modelo de gestión de los recursos humanos que tienen las organizaciones debe evolucionar, no sólo con el fin de obtener un beneficio económico de manera inmediata, sino considerando que los distintos trabajadores pueden responder de diversas maneras a unas mismas políticas, en función de sus intereses y preferencias, así como de sus formas de entender la realidad y el trabajo.

1. Descripción del ámbito de estudio

En el entorno en el que se desenvuelven las organizaciones actualmente, las estructuras clásicas de gestión de recursos humanos dejan de tener sentido, ya que las empresas tradicionales, interesadas únicamente en obtener beneficios económicos de manera inmediata tendrán dificultades para competir en un entorno dinámico que cada vez cambia más rápidamente.

Los modelos clásicos de gestión de los recursos humanos partían de la base de que todos los trabajadores formaban parte de una única categoría de manera homogénea. En otras palabras, todos los trabajadores eran iguales.

De esta manera, en estos modelos clásicos, hechos como la remuneración, formación o motivación eran diseñadas sin tener en cuenta a los trabajadores, es decir, se aplicaban estas prácticas por igual en todos ellos sin considerar que los trabajadores pueden responder ante unas mismas políticas de manera diferente. Esto se debe a que existen ciertos factores, como intereses personales, preferencias, ideas, formas de entender la realidad y el trabajo, etc, que son diferentes en cada persona. Supongamos una empresa que decide dar un mayor peso al salario variable, ¿obtendría dicha empresa la misma respuesta por parte de todos sus empleados? Por ejemplo, si tuviéramos un empleado de 50 años, con una familia a su cargo y otro empleado mucho más joven, sin compromisos y que acaba de comenzar su carrera profesional, ¿La respuesta sería la misma? Obviamente, el empleado joven aceptará estas condiciones más fácilmente que el empleado de más edad, y aunque esta respuesta es aparentemente sencilla y muestra la existencia de diferencias entre los empleados de la empresa, estos modelos no contemplaban la posible heterogeneidad dentro de los equipo de trabajo.

Los nuevos requisitos que imponen debido a los nuevos entornos caracterizados por su dinamismo, complejidad y una creciente diversidad dentro de los mercados de trabajo, han hecho que estos modelos resulten ineficientes y se rompan. ¿Tiene sentido plantear un mismo sistema de compensación para toda la empresa si esta cuenta con trabajadores con intereses distintos?, ¿serán iguales los intereses formativos de todos los empleados de la organización?, ¿responderán por igual todos los empleados ante unos mismos incentivos?, ¿estarán todos los empleados dispuestos a sacrificar, de igual manera, el tiempo que dedican a ocio o a su familia por promocionar en su puesto u obtener un incremento salarial?...

Todas las cuestiones planteadas anteriormente ponen de manifiesto la necesidad de modificar y adaptar el sistema de gestión de los recursos humanos a las nuevas necesidades del entorno.

Dentro de este ámbito, en nuestro caso, nos centraremos en los modelos de gestión de los equipos de trabajo de desarrollo de software. Sin embargo, para poder analizar y estudiar la diversidad existente en los equipos de trabajo y como puede ser ésta gestionada y aprovechada por parte de las empresas, es necesario conocer previamente cuáles son los motivos por los cuales se ha producido esta nueva situación del entorno de trabajo.

Cabe destacar tres grandes factores:

a) Globalización de la economía:

En la actualidad estamos sufriendo una mundialización de la economía. Este hecho supone una libre circulación de capitales, bienes y servicios justo con una rápida globalización de las economías que está provocando que los mercados de trabajo cada vez sean más competitivos. Existe una tendencia hacia la liberalización del comercio, de manera que las barreras de protección de los mercados se eliminan; hacia una mayor libertad y facilidad de circulación del capital para pasar fronteras; hacia impulsar el progreso de las comunicaciones, etc. Por lo que ante esta nueva situación, la exigencia de nuevos y más altos requerimientos de calidad de los productos y servicios ofertados es mucho mayor. Esto es así debido a que hay que competir en un mercado cada vez más abierto por lo que los precios deben ajustarse teniendo además los productos y servicios ofertados tienen ciclos de vida útil más cortos.

Esta nueva estructura de la economía, supondrá una modificación en los requisitos de competencias que anteriormente las empresas exigían a los trabajadores. Por lo tanto factores como la transferibilidad, flexibilidad o versatilidad y polivalencia de competencias profesionales serán mucho más valorados y tenidos en cuenta. Para que el producto o servicio de una organización sea innovador, eficaz y de calidad, se requiere de personal cualificado, y esta cualificación no puedes ser conseguida únicamente mediante la experiencia laboral. Dada esta situación, se plantean cuestiones que difícilmente encuentran respuestas claras y concretas. Por ejemplo, ¿cuáles serán las cualificaciones, competencias, capacidades, etc que aseguren a una organización que su productividad y crecimiento se verá reforzado?

Ante estas circunstancias, tanto la enseñanza y la formación profesional se presentan como elementos indispensables para mejorar la competitividad y productividad de las economías y la eficacia del mercado de trabajo.

b) Nuevas estructuras de trabajo

Actualmente, debido a las nuevas condiciones de competitividad del mercado, se han producido nuevas estructuras de trabajo caracterizadas por un desarrollo del trabajo en equipo, evitando las jerarquías, vinculándose con otras normas de gestión de los recursos humanos y participando en actividades ajenas. De esta manera, las organizaciones empresariales se ven en la situación de seleccionar adecuadamente los

propietarios, directivos y profesionales, de establecer y entablar buenas alianzas con otras empresas, de desarrollar y aplicar planes de formación continua, etc.

Debido a la reducción de las jerarquías, a los trabajadores se les exige una mayor flexibilidad y adaptabilidad, más colaboración y trabajo en equipo así como una mayor autonomía individual, mayor cualificación y formación en distintos ámbitos, reciclaje y formación permanente, etc.

c) Importancia del sector servicios y avance de las tecnologías

Por último, en todo el mundo, las nuevas tecnologías de la información y las comunicaciones están generando una auténtica revolución. La tecnología se ha convertido en un factor competitivo y requiere que las empresas se adapten a ese nuevo marco mediante nuevas organizaciones y servicios ofertados a los clientes.

El continuo contacto que las empresas del sector servicios relacionadas con el desarrollo de software tienen con el cliente hace que éstas tengan que adaptarse a las necesidades de éstos, de manera que la plantilla de profesionales que cada empresa tenga tiene que estar suficientemente formada y preparada para saciar cualquier exigencia por parte del cliente.

2. Delimitación de la solución

El proyecto, *Gestión multi-variable de equipos multi-disciplinarios e internacionales de equipos y proyectos de desarrollo de software*, nace ante la necesidad de gestionar la diversidad en los equipos de trabajo, ya que se está convirtiendo en uno de los principales retos a los que tiene que enfrentarse una organización, debido a causas como las mencionadas anteriormente: un aumento de la globalización, cambios en los mercados de trabajo, nuevas tecnologías, etc.

En la actualidad, tener equipos de trabajo heterogéneos no es una obligación, sin embargo, es una fuente de ventajas competitivas, debido en parte a dos razones:

- Cada cliente es diferente del resto, de manera que esta diversidad de clientes puede ser mejor gestionada y atendida teniendo también diversidad dentro del equipo de trabajo que se podrá comunicar con el cliente de la mejor manera posible, así como ofrecerle y suministrarle el proyecto que mejor se adapte a sus requisitos.
- La diversidad dentro del equipo de trabajo generará un amplio rango de ideas y soluciones para gestionar los requisitos y problemas que se planteen.

Lazear (1998a, 1998b) [1] [2] afirma que un equipo de trabajo en el que hay presente una gran diversidad dará lugar a una gran productividad por parte de este equipo siempre que tres factores estén presentes:

- Los miembros del equipo deben tener diferentes capacidades técnicas, habilidades y conocimientos, de manera que entre ellos se puedan complementar para alcanzar la mejor solución.
- Estas capacidades técnicas, habilidades y conocimientos deben ser relevantes e importantes para el desarrollo del proyecto, ya que si no afectan a la

producción y desarrollo del proyecto estas no serán valoradas ni tenidas en cuenta.

- La comunicación es imprescindible en un equipo de trabajo para que fluya el intercambio de ideas y conocimientos entre ellos. Un incremento de los costes de comunicación entre los miembros del equipo reduce los beneficios obtenidos de la diversidad de conocimientos y capacidades de los miembros.

Estos tres factores destacan que al menos dos aspectos deben ser tenidos en cuenta a la hora de analizar la diversidad en los equipos de trabajo: (1) la diversidad en las capacidades técnicas, habilidades y conocimientos dentro de los miembros del equipo; y (2) la diversidad en otros factores que influyen en la comunicación entre los miembros del equipo. Por lo tanto, lo que el estudio de Lazear plantea es que los equipos de trabajo productivos deben tener una amplia diversidad en cuanto a capacidades técnicas, habilidades y conocimientos se refiere, pero la menor diversidad posible en otras dimensiones como pueden ser factores demográficos que reduzcan los costes de comunicación o lo que ellos denominan “costes culturales”.

3. Estructura de la memoria

El formato del presente proyecto, está estructurado en seis partes que se describen brevemente a continuación.

La primera parte del documento, denominada introducción y objetivos, está formada por los capítulos 1 y 2. El primer capítulo implica una introducción formada por la descripción del ámbito de estudio, así como la descripción de su problemática y la delimitación de la solución propuesta. El segundo capítulo describe los objetivos planteados para el desarrollo de este proyecto.

La segunda parte del documento es el estado del arte, que comprende los capítulos 3, 4, 5 y 6. En esta sección trataremos de estudiar, analizar y describir todos los diferentes factores, algoritmos, metodologías, procesos, etc, que van a tener lugar o formar parte más delante de nuestro sistema. En el capítulo 3, analizaremos y describiremos todos los factores que pueden influir notablemente en nuestro proyecto, es decir, que influyen en la productividad que un determinado equipo de trabajo puede alcanzar durante el desarrollo de un proyecto de software. En los capítulos 4 y 5, realizaremos una descripción de las redes de neuronas y de los algoritmos genéticos respectivamente, para tener los conceptos necesarios para comprender el sistema. Y por último, en el capítulo 6, expondremos la metodología de desarrollo que seguirán los equipos formados por el sistema para alcanzar esta mayor productividad.

La tercera parte es el modelo conceptual del sistema que está formado por los capítulos 7 y 8. En el capítulo 7, se explica cómo se relacionan todos los factores que influyen en la productividad de un equipo, mientras que en el capítulo 8, se describen los distintos requisitos de software del sistema a desarrollar.

La arquitectura del sistema se desarrolla en la cuarta parte del documento que está formado por el capítulo 8. En este capítulo, se aborda la arquitectura del sistema desarrollado y la distribución de la funcionalidad del sistema en esta arquitectura.

La quinta parte de la documentación implica los capítulos 9, 10 y 11. Entre estos dos capítulos se describe toda la solución propuesta al sistema planteado. Por un lado, en el capítulo 9, se describe todo el sistema: tanto el procedimiento realizado para desarrollarlo, como las justificaciones a las decisiones tomadas. Por otro lado, en el capítulo 10, mostramos a través de unas imágenes la interfaz creada para nuestro sistema. Además, este apartado consta del capítulo 11 en el que se presentan las futuras vías de estudio del problema planteado y posibles mejoras futuras del sistema.

Finalmente, la sexta parte es en la que se encuentra el capítulo 12, la bibliografía. En este capítulo se citan la lista de fuentes de consulta utilizadas para llevar a cabo el presente proyecto.

Capítulo 2 | Objetivos

A lo largo de la historia, el trabajo en equipo cada vez ha ido cogiendo más fuerza. Ya por los años 30, Henry Ford afirmaba 118:

"Reunirse en equipo es el principio.
Mantenerse en equipo es el progreso.
Trabajar en equipo asegura el éxito".

El proyecto, *Gestión multi-variable de equipos multi-disciplinarios e internacionales de equipos y proyectos de desarrollo de software*, nace con un objetivo claro, proponer un modelo que nos ayude a obtener la mejor combinación posible de miembros de un equipo consiguiendo así formar el equipo de trabajo que genere una mayor productividad y obtenga el mejor resultado para un determinado proyecto.

Lo que se pretende con este proyecto es encontrar la manera adecuada para relacionar todas las variables que intervienen a la hora de trabajar en equipo. Para poder llevarlo a cabo, establecemos los siguientes objetivos:

- **Objetivo 1:** Conocer la situación actual a la que se enfrentan los equipos de trabajo de desarrollo de software, así como los principales factores que afectan a la hora de desarrollar un proyecto dentro del equipo de trabajo
- **Objetivo 2:** Analizar y describir los factores más significativos que intervienen en el trabajo en equipo.
- **Objetivo 3:** Establecer un método de medición para cada factor destacado en el punto anterior de manera que sea medible y evaluable por el sistema.
- **Objetivo 4:** Diseñar un sistema que sea capaz de gestionar y relacionar de manera óptima todos los factores que le afectan.
- **Objetivo 5:** Validación del sistema mediante la validación de los resultados obtenidos.

PARTE II:
ESTADO DEL ARTE

Capítulo 3 | Equipos de trabajo

El hombre es un ser social que necesita mantener contactos en la empresa, por lo que el trabajo en equipo está sustituyendo cada vez más el trabajo individual. Pero para comprender bien cómo gestionar y componer los equipos de trabajo a la hora de desarrollar un proyecto, primero necesitamos saber qué es un equipo de trabajo y qué diferencia hay con los grupos de trabajo.

Un grupo de trabajo es un conjunto de personas que dentro de una organización tienen una formación similar y desempeñan una labor similar, ante la que responden de manera individual, es decir, cada uno realiza su trabajo y no dependen del trabajo de sus compañeros. Por ejemplo, un grupo de anestesistas dentro de un hospital realizan la misma tarea y tienen en común al jefe de la sección, sin embargo, cada uno realiza y responde por su trabajo, y este no se ve influido por el de los demás.

Por otro lado, un equipo de trabajo es un conjunto de personas que dentro del proyecto en el que participan responden todos por igual como conjunto ante la labor desempeñada. Dentro de un equipo de trabajo hay un jefe de equipo con una serie de colaboradores elegidos en función de sus conocimientos, habilidades, capacidades personales y técnicas, etc.

Al contrario que en un grupo de trabajo, los colaboradores o miembros del equipo de trabajo pueden tener categorías laborales diferentes pero todos funcionan al mismo nivel y aunque cada uno de ellos realiza una parte concreta del proyecto, todas las partes del proyecto son complementarias y dependen unas de otras, por lo que se requiere coordinación, cohesión, etc, entre todos ellos.

Una vez que hemos comprendido qué es un equipo de trabajo y qué diferencia tiene con un grupo de trabajo, vamos a profundizar en cómo son y cómo se gestionan los equipos de trabajo en las organizaciones actuales.

Dentro de un equipo de trabajo, cabe destacar las siguientes características:

- Integración: las tareas realizadas por diferentes miembros del equipo deben ser integrables y realizadas de manera armónica.
- Responsabilidad compartida: todos los miembros de un equipo tienen la misma responsabilidad en el trabajo desarrollado.
- Coordinación: todas las actividades desarrolladas por un equipo deben realizarse de manera coordinada habiendo relación entre todos los miembros que lo forman.
- Mismo objetivo: los programas planteados en equipo debe ir dirigidos hacía el mismo objetivo.

Sin embargo, para convertir transformar un grupo de trabajo en un equipo y alcanzar estas características es necesario por parte de las organizaciones facilitar un tiempo de adaptación en el cual se consigan los siguientes conceptos:

- Cohesión y unión: es una condición indispensable para poder ser miembro de un equipo.

- Roles y normas: todos los equipos deben asignar roles y normas a sus miembros. Las normas son reglas que gobiernan el comportamiento de los miembros del equipo a tenerse a los roles explícitamente definidos permite al equipo realizar las tareas de un modo eficiente.
- Comunicación: una buena comunicación interpersonal es imprescindible para el desarrollo de cualquier tipo de tarea dentro del equipo de trabajo.
- Definición de objetivos.
- Interdependencia positiva: sus miembros se necesitan unos a otros, interactúan diariamente y cada uno aprende de los demás compañeros.

Una vez alcanzados estos conceptos podemos hablar de equipo de trabajo. Aparentemente, todo lo que suponen los equipos de trabajo a las empresas son ventajas y beneficios, sin embargo, en la actualidad, las empresas con el fin de crear nuevas estrategias de negocio, así como desarrollar productos y servicios innovadores para sus clientes, están creando equipos de trabajo **multidisciplinares** dónde entre los miembros del equipo puede llegar a existir una gran diversidad.

De acuerdo a investigaciones de IDC y estudios realizados por QSMA (Quantitative Software Management Associates) podemos afirmar que: [3]

- Los proyectos organizados con equipos multidisciplinares entregan los productos con una productividad y calidad 16% mejor que el resto de la industria.
- Los proyectos desarrollados aplicando una metodología ágil, cuya base fundamental son los equipos multidisciplinares, producen y entregan sus productos un 37 % más rápido que el resto de la industria.

Otro ejemplo, de la importancia de los equipos multidisciplinares en la actualidad es EVER TEAM, que está considerado Líder en Europa, reconocido por la satisfacción de sus Clientes y los grandes analistas del mercado como Gartner, CMS Watch – The Real Story o Forrester entre otros, y que basa su estrategia de negocio en formar equipos multidisciplinares para satisfacer las necesidades y estrategia del cliente. [4]

Pero, ¿qué es la diversidad en un equipo de trabajo? Podría definirse la diversidad como una propiedad de los grupos de trabajo que mide la heterogeneidad de sus miembros en relación a una serie de características personales como por ejemplo, el género, la raza, el nivel socio-económico, personalidad, cultura, nacionalidad, edad, educación, formación, etc, así como del impacto que tiene para la dinámica y la efectividad de los equipos de trabajo.

Aparentemente la diversidad puede verse como un hecho positivo para el equipo de trabajo ya que promueve diferentes y nuevas ideas y perspectivas, genera una mayor cantidad de opiniones (“brainstorming”), etc, todo lo cual puede dar lugar a soluciones mucho más creativas e innovadoras para los clientes. Sin embargo, esta definición como tal, no nos permite aclarar dos cuestiones importantes: por un lado, el tipo de atributos o características que estamos valorando; por otro lado, las posibles relaciones que pueden darse entre las diferentes características para generar una mayor o menor productividad. ¿Funcionará igual un equipo que sea muy heterogéneo en cuanto a la nacionalidad de

sus miembros que otro con un grado muy elevado de diversidad de género?, ¿y uno con una gran diversidad de edad frente a otro con una gran diversidad étnica?

Por tanto, dentro de la diversidad no todo es positivo, ya que la diversidad en los equipos de trabajo puede generar una falta de integración social por parte de un miembro del equipo, o una falta de entendimiento debido al idioma, etc.

Conocer los mecanismos a través de los cuales la diversidad ejerce este efecto negativo sobre el grupo de trabajo y las condiciones en las que se transforma en una ventaja para el equipo es la clave del éxito. Por este motivo, parece imprescindible destacar cuáles son las características o atributos en los que los miembros de un equipo pueden diferir y que influyen en la productividad y efectividad de un equipo.

Como hemos mencionado en los primeros apartados del documento cabe destacar dos principales focos que afectan a la productividad del equipo.

1. Comunicación:

Dentro de este ámbito, los principales factores que influyen y afectan a la comunicación entre los miembros del equipo suelen ser atributos o características innatas de la persona como por ejemplo, la edad, nacionalidad, cultura,…aunque también hay otros como el país de residencia que también puede influir.

Los principales factores o atributos que afectan a la comunicación dentro de un equipo de trabajo son:

1.1 Cultura:

El término cultura, (del lat. cultūra) es y ha sido utilizado en muchos ámbitos a lo largo de la historia. Actualmente, según la RAE, la cultura es el “conjunto de modos de vida y costumbres, conocimientos y grado de desarrollo artístico, científico, industrial, en una época, grupo social, etc.”

Este factor está vinculado a la nacionalidad de cada individuo y quizás sea uno de los más determinantes a la hora de llevar a cabo el desarrollo de un proyecto de software, ya que para alcanzar un resultado óptimo el entendimiento entre todos los miembros del equipo debe ser total. Hay estudios que afirman que “las personas de diferentes nacionalidades poseen distintos programas mentales, a partir de los cuales construyen percepciones diferentes sobre lo que sucede. Estas diferencias en las percepciones son una fuente potencial de conflictos que pueden dificultar los acuerdos y el entendimiento entre las personas”.

Según Geert Hofstede (1991)118 la cultura implica una forma determinada de resolver problemas: personas de diferentes culturas resuelven problemas similares de maneras diferentes. Pero la cultura no es solo una forma de entender el mundo, sino también de construirlo. Esta distinción es importante porque remarca el rol activo que tiene la cultura en la construcción de la realidad, de aquello que vemos y dejamos de ver, de aquello que interpretamos y, principalmente, de aquello que hacemos.

Hofstede, identificó una serie de dimensiones para clasificar las diferentes culturas nacionales. Las cuatro dimensiones identificadas por el autor son:

- **Índice de distancia al poder (PDI):**

El índice de distancia al poder se refiere al grado de aceptación, en los miembros menos poderosos de una cultura, de las diferencias de poder o la desigualdad. El PDI será mayor en la medida en que dichas diferencias en estructuras sociales o jerárquicas (como empresas, instituciones o familias) sean muy amplias y generalmente aceptadas.

- **Índice de Colectivismo vs individualismo:**

El IDV define el nivel en que los individuos se integran en la sociedad y el sentimiento de pertenencia al grupo. En una sociedad con alto IDV, por ejemplo, los individuos tienden a preocuparse de sí mismos y de su familia más cercana, mientras que en una sociedad muy colectivista, los lazos grupales son más amplios y la unidad familiar es mucho más extensa.

- **Índice de Feminidad vs Masculinidad:**

Este indicador define la tendencia de una cultura hacia patrones de conducta de una mayor masculinidad o femineidad. El estudio de Hofstede revelaba que las sociedades masculinas eran más asertivas y competitivas frente a las femeninas, generalmente más modestas y empáticas. En las sociedades masculinas hay una mayor brecha en cuanto a los valores masculinos y femeninos, y las mujeres tienden a ser más competitivas y asertivas. A mayor valor de este índice más masculinidad.

- **Índice de control de la incertidumbre (UAI):**

El UAI trata de la aceptación de la sociedad de la incertidumbre y la ambigüedad frente a una verdad absoluta. Según Hofstede, un país con alto UAI tratará de evitar riesgos, situaciones desestructuradas, o que se salgan de lo habitual. Dichos países son más emocionales, suelen reforzar la seguridad con leyes estrictas, y a un nivel filosófico y religioso, creen en una verdad absoluta. Por contra, los países con bajo UAI suelen ser más reflexivos, tolerantes y relativistas.

En función de estas dimensiones podremos comprender las diferencias culturales entre miembros provenientes de diferentes culturas. Los valores no son índices absolutos que reflejan con exactitud la cultura de un país, pero nos servirán de apoyo a la hora de evaluar las diferentes nacionalidades y las relaciones entre ellas.

1.2 Idioma

En el caso de este factor es evidente la relación que guarda con los procesos de comunicación entre los miembros del equipo, ya que sin un idioma en común entre todos ellos mediante el cual expresar sus ideas y relacionarse, el coste de comunicación se verá aumentado notablemente dificultando en gran medida el desarrollo del proyecto.

Supongamos que formamos parte de un equipo de trabajo con una gran diversidad de habilidades y conocimientos pero cada uno procedemos de un país de origen distinto con distintos idiomas nativos, si tuviéramos que preguntar al experto del equipo en redes de comunicación alguna duda pero ninguno hablamos su idioma, ¿cómo lo hago? ¿Retrasaría el desarrollo del proyecto?

Este ejemplo deja de manifiesto que un idioma común es imprescindible para obtener el mejor desarrollo del proyecto.

1.3 Edad

La diversidad edad entre los miembros de un equipo es un factor que puede llegar a afectar notablemente al desarrollo de software y está altamente relacionado con la experiencia profesional de las personas.

Con una diferencia de edad de 50 años entre empleados existente en algunas organizaciones, es inevitable reconocer que cada generación tiene un conjunto diferente de expectativas, necesidades y valores a la hora de trabajar.

Diversos estudios sobre la influencia de la edad en el trabajo en equipo han sido realizados, obteniendo resultados muy dispares. Por un lado, por ejemplo, algunos investigadores como Knight, D., Pearce, C. L., Smith, K. G ., Olian, J . D., Sims, H . P., Smith, K. A., & Flood, P. (1999) [6], afirmaron que una amplia diversidad entre los miembros de un equipo suponía una gran dificultad para trabajar y llegar a acuerdos. Sin embargo, otros investigadores como son Pelled, L. H., Eisenhardt, K. M., & Xin, K . R. (1999), [7] obtuvieron resultados opuestos, y afirmaron que los equipos de trabajo con una amplia diversidad de edad experimentaban mejores resultados, ya que esta diferencia de edad evitaba gran variedad de conflictos y a su vez genera un mayor conjunto de puntos de vista y soluciones.

2. Desarrollo del proyecto y alcance de objetivos

En este ámbito, los principales factores que influyen y afectan a la comunicación entre los miembros del equipo suelen ser atributos o características adquiridas a lo largo del tiempo por parte de la persona y que pueden verse modificadas como por ejemplo, la experiencia y motivación laboral, estudios y capacidades técnicas, personalidad, etc.

Los principales factores o atributos que afectan al buen desarrollo del proyecto dentro de un equipo de trabajo son:

2.1 Motivación

La motivación laboral de cada miembro del equipo es un factor muy importante que puede afectar al desarrollo y evolución de un proyecto de desarrollo de software. A la hora de elegir los miembros que formen un equipo de desarrollo será importante conocer el nivel motivacional de cada uno de ellos. Hay estudios que afirman que la

motivación laboral es el resultado de multiplicar tres factores: la valencia, la instrumentalidad y las expectativas; esta afirmación es conocida como la teoría de las expectativas o teoría VIE propuesta por Vroom, V. H. (1964) [8]. De esta manera, una persona estará motivada en función del grado en que ella crea que su esfuerzo (a) va a producir el resultado esperado (Expectativa), este resultado (b) va a ser valorado y recompensado de alguna manera (Instrumentalidad), y (c) que además la persona tenga deseo de alcanzar ese resultado (valencia).

- **Valencia:**

Demuestra el nivel de deseo de una persona por alcanzar determinada meta u objetivo. Este nivel de deseo varía de persona a persona y en cada una de ellas puede variar a lo largo del tiempo, estando condicionada por la experiencia de cada individuo. El rango de valores que puede admitir la valencia en esta ecuación matemática es entre –1 y 1. Cuando una persona no quiere llegar un determinado resultado, por ejemplo ser despedido de su trabajo, el valor adoptado es –1; cuando el resultado le es indiferente, como por ejemplo recibir una compensación en dinero, el valor es 0; y cuando la persona pretende alcanzar el objetivo, por ejemplo obtener un ascenso, su valor será de 1.

- **Instrumentalidad:**

Es la probabilidad que la persona estima en función de que una vez realizado el proyecto, la organización lo valore y reciba su recompensa. El valor de la instrumentalidad será entre 0 y 1. Por ejemplo, si un empleado ve que un buen desarrollo y resultado de un proyecto supone un aumento de sueldo, la instrumentalidad tomará valor 1. Sin embargo, si el empleado no ve relación entre un buen desarrollo y resultado de un proyecto y un aumento de sueldo, la instrumentalidad tomará valor 0.

- **Expectativa:**

Representa la convicción de una persona en que el esfuerzo depositado en su trabajo producirá el efecto deseado. Su valor varía entre 0 y 1, ya que, la expectativa es la probabilidad de ocurrencia del resultado deseado. Las expectativas dependen en gran medida de la percepción que tenga la persona de sí misma, si la persona considera que posee la capacidad necesaria para lograr el objetivo le asignará valor 1, en caso contrario le asignará valor 0.

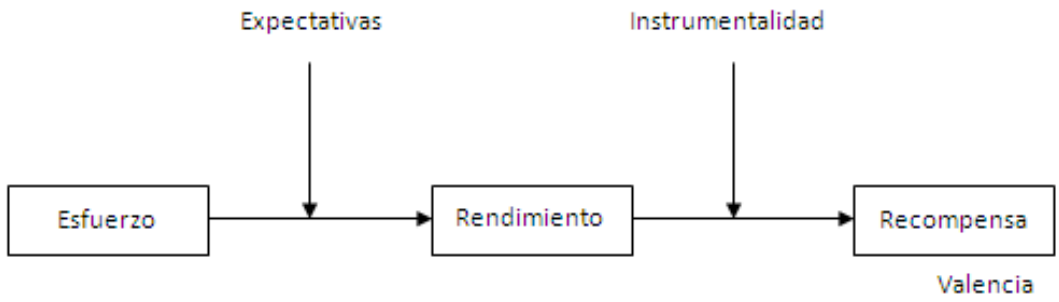

Ilustración 1 - Motivación laboral

2.2 Experiencia

Los trabajadores pueden ver enriquecido su capital humano, además de tener los conocimientos y las aptitudes necesarias, por haber experimentado previamente situaciones laborales o procesos de trabajo a los que posteriormente pueden enfrentarse en su equipo de trabajo. Por lo tanto, la existencia dentro de un equipo de trabajo de personas con distintos grados y tipos de experiencia, fortalecerá, como veremos en los siguientes capítulos, el rendimiento del equipo que, como consecuencia, será mucho más capaz de tomar decisiones y resolver problemas complejos. (Reagans et al. 2005; Espinosa et al. 2007; Huckman et al. 2009) [9] [10] [11].

2.3 Personalidad

La personalidad de cada miembro del equipo es también un factor que puede influir de manera significativa a la hora de llevar a cabo un proyecto.

Personalidad (Del lat. personae: máscara). [12]

1. f. Diferencia individual que constituye a cada persona y la distingue de otra.

2. f. Conjunto de características o cualidades originales que destacan en algunas personas.

3. f. Persona de relieve, que destaca en una actividad o en un ambiente social.

4. f. Inclinación o aversión que se tiene a una persona, con preferencia o exclusión de las demás.

5. f. Dicho o escrito que se contrae a determinadas personas, en ofensa o perjuicio de las mismas.

6. f. Der. Aptitud legal para intervenir en un negocio o para comparecer en juicio.

7. f. Der. Representación legal y bastante con que alguien interviene en él.

8. f. Fil. Conjunto de cualidades que constituyen a la persona o sujeto inteligente.

La personalidad de una persona es única e individual, no hay dos personalidades iguales, ya que dos personas pueden tener rasgos parecidos pero en conjunto no todos ellos son iguales. La relación entre distintas personalidad dentro de un equipo de trabajo puede dar lugar a un resultado mejor o peor del proyecto. Sin embargo, aunque todos los factores que influyen en la personalidad son importantes, a la hora de desarrollar un proyecto de software cabe destacar un rasgo fundamental que es la capacidad de liderazgo. Un equipo formado, por ejemplo, por 5 miembros de grandes capacidades técnicas pero que cuyas personalidades tienden a liderar, podría provocar un resultado nefasto del proyecto, debido al choque de personalidades y a la necesidad de todos de estar por encima del resto. Sin embargo, otro equipo de 5 miembros con las mismas

capacidades técnicas pero que esta vez está formado por solo una personalidad que tiende al liderazgo será mucho más fácil que obtenga un mejor resultado.

Este factor, indirectamente, también afectará a los procesos de comunicación entre el equipo, según la forma de liderar del o de los líderes del equipo.

2.4 Capacidades técnicas y conocimiento

Todo el modelo que estamos construyendo tiene como base fundamental medir la productividad o el trabajo en equipo de equipos desarrolladores de software, por los tanto, las capacidades, habilidades y conocimientos que vamos a valorar son los relacionados con este campo. Algunos de los requisitos indispensables que cada miembro debería dominar o al menos conocer son:

- Gestión de requisitos
- Estimación y priorización de requisitos
- Gestión de reuniones
- Resolución de problemas técnicos
- Capacidad de autoaprendizaje
- Capacidad para trabajar en equipo (estableciendo reglas, pautas, etc.) y habilidades para relaciones interpersonales
- Capacidad de organización y planificación
- Programación
- Manejo de diferentes plataformas
- Etc.

3. Otros factores

Además, existen otra serie de factores que influyen tanto en la comunicación como en el desarrollo del proyecto y su resultado:

3.1 Número de miembros del equipo

Este factor no depende tanto de las características individuales de cada miembro, sino del grado de complejidad del proyecto así como de sus características, en función de los cuales necesitaremos más o menos miembros en el equipo de trabajo.

Cabe destacar que un menor número de miembros facilitará la comunicación pero, a su vez, supondrá un aumento en el número de tareas a realizar, un mayor tiempo de dedicación y por lo tanto, aumentará la dificultad para desarrollar el proyecto. Por lo tanto, resulta evidente la necesidad de alcanzar un punto medio que facilite tanto la comunicación como el desarrollo de tareas dentro del proyecto.

3.2 Complejidad y duración del proyecto

Otro factor a tener en cuenta es la duración del proyecto, ya que si un proyecto es a corto plazo quizás se requiera más experiencia por parte de los miembros del equipo ya que no hay mucho tiempo para aprender y por tanto, esto podrá influir en el número de miembros y en el tipo de miembros que formarán el equipo.

Capítulo 4 | Redes de Neuronas

1. Introducción

Las Redes Neuronales Artificiales o RNA (*Artificial Neural Networks o ANN*) están fundamentadas en las redes neuronales biológicas del cerebro humano. Las RNA las forman elementos que tratan de simular el comportamiento de la neurona biológica en sus funciones más comunes, estando, estos elementos, organizados de manera similar a la que se puede encontrar en el cerebro humano [13].

La neurona es la unidad fundamental del sistema nervioso y en particular del cerebro. Cada neurona es una simple unidad procesadora que recibe y combina señales desde y hacia otras neuronas. El cerebro consiste en uno o varios billones de neuronas densamente interconectadas. La salida de la neurona o axón se ramifica y está conectada a las entradas o dendritas de otras neuronas a través de uniones llamadas sinapsis. La eficacia de la sinapsis es modificable durante el proceso de aprendizaje de la red.

Estas RNA, por tanto, presentan una cierta analogía con el cerebro.

- Aprender*:* son capaces de cambiar su comportamiento en función del entorno y de su experiencia previa. Se les muestra un conjunto de entradas (inputs) y ellas mismas se ajustan para producir unas salidas (outputs) consistentes.

- Generalizar*:* son capaces de generalizar este aprendizaje gracias a su estructura automáticamente, siendo capaces, dentro de un margen, de ofrecer soluciones o respuestas a conjuntos de entrada que presentan mínimas variaciones debido a los efectos de ruido.

- Abstraer*:* son capaces a partir de un conjunto de entradas, que a priori no presentan aspectos comunes, abstraer un modelo común.

Fundamentalmente el proceso que sigue una neurona biológica es el siguiente: mediante unos datos de entrada la neurona es excitada, y una vez que ésta alcanza cierto nivel o umbral, la neurona se activa y envía una señal al axón. La combinación del cuerpo de la neurona con las dendritas (vía de entrada a la neurona) ayudan a generar la señal de salida de la neurona. La conexión o interacción entre neuronas es denominada sinapsis. La *Ilustración 2* muestra cómo es una neurona biológica.

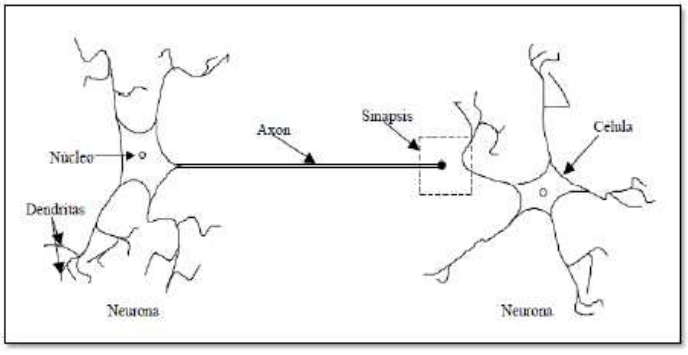

Ilustración 2 - Composición Red de neurona humana

2. Red neuronal artificial

Una red neuronal artificial consiste en un conjunto de unidades elementales conectadas de una forma concreta que dan lugar a un modelo computacional inspirado, como se ha mencionado anteriormente, en redes neuronales biológicas. Las principales características de las redes de neuronas artificiales son: robustez y adaptabilidad, tolerancia a fallos, etc. [14]

Además del modelo que proporciona la red, es importante conocer las formas en que se conectan estos elementos procesadores. Generalmente los elementos de la red están organizados en grupos denominados capas: *una capa de entrada* que es donde se presentan los datos a la red, *otra de salida* que mantiene la respuesta de la red a una entrada, y en el caso de existir, una o varias *capas ocultas*.

Cada neurona recibe un valor de entrada, que es transformado según una función específica denominada *función de activación*. Y el resultado de esta función de activación pasa a ser la salida de la neurona.

Las neuronas se conectan entre sí según una determinada arquitectura, de manera que la salida de cada neurona se propaga por igual por estas conexiones hasta las neuronas de destino. Cada conexión tiene un *peso* asociado que pondera la señal que viaja por dicha conexión. De esta manera, la entrada de cada neurona es el resultado de sumar las salidas de las neuronas conectadas a ella, multiplicadas cada una de éstas por el peso de la respectiva conexión.

Así pues, una red de neuronas artificial puede verse como un grafo cuyos nodos tienen funcionamiento similar, los cuales propagan la información a través de las distintas conexiones.

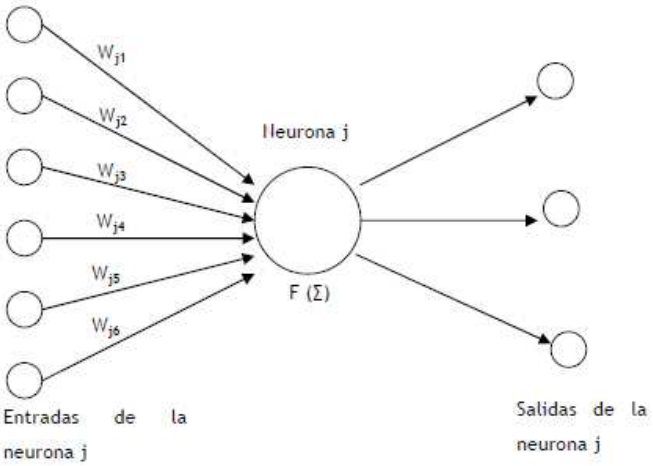

Ilustración 3 – Funcionamiento Red Neuronas Artificial

Las RNA son sistemas de aprendizaje basadas en datos que son utilizados como patrones y deben ser previamente entrenadas para poder generar un modelo fiable. Por tanto, la capacidad de una RNA de resolver un problema está muy ligada a los patrones de datos utilizados durante su fase de aprendizaje.

2.1 Elementos básicos que componen una RNA

La estructura de una red se distribuye en capas en cada una de las cuales se distribuyen todas las unidades que la forman. Esta red está caracterizada por las conexiones de unas unidades con otras [14]. Las tres capas que existen en una RNA son:

- Capa de entrada: las neuronas de esta capa a partir de los datos de los patrones de entrada generan una salida que se convierte en la entrada de las neuronas de la siguiente capa. Es decir, las entradas al sistema se reciben en esta capa.

- Capa oculta o intermedia: estas neuronas pueden distribuirse en una o varias capas ocultas, y reciben como señal la salida de la capa de entrada y, de nuevo, su salida es la entrada de la siguiente capa. En esta capa se produce toda la función de la red, es decir, se extrae procesa y memoriza la información que se recibe. El procesamiento de esta información se basa principalmente en los esos de cada una de las conexiones entre neuronas.

- Capa de salida: estas neuronas reciben como entrada la salida de las neuronas de la capa oculta. Al contrario de la capa de entrada, la salida de estas neuronas no son la entrada de ninguna otra capa. Por lo tanto, de éstas neuronas obtenemos el resultado final al problema planteado.

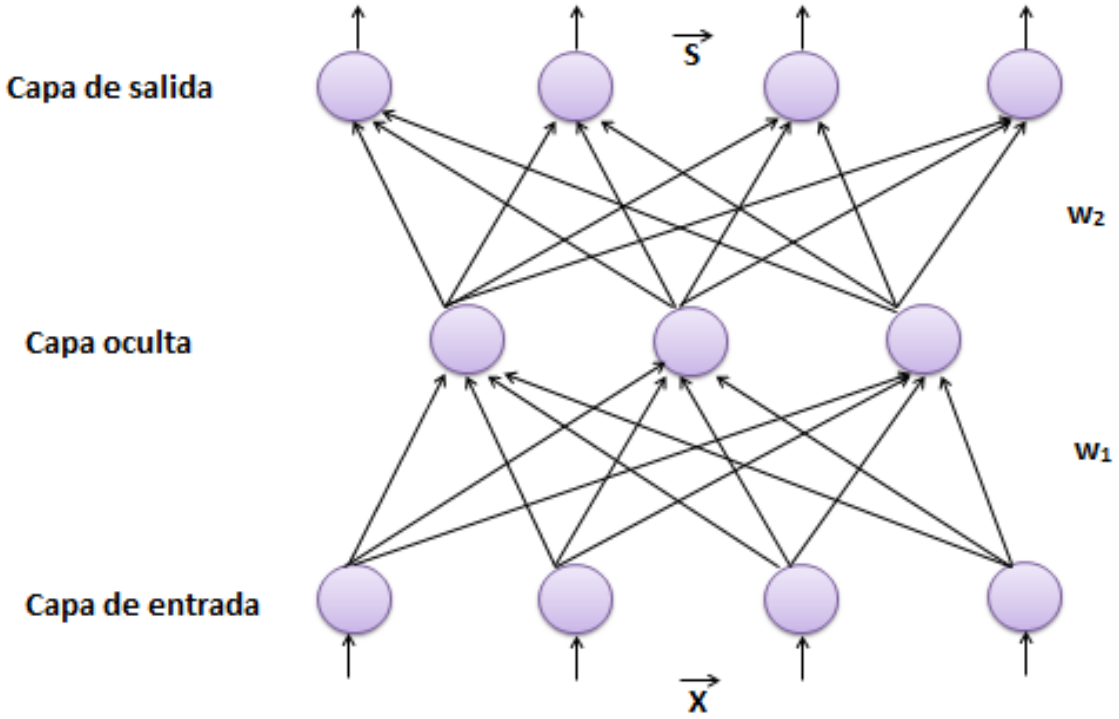

Ilustración 4 - Componentes Red Neuronas Artificial

2.1.1 Función de entrada

La función de entrada, o también denominada función de propagación o ponderación, transforma las entradas que recibe el sistema. Algunas de las funciones de entrada más utilizadas y conocidas son:

- Suma: todos los valores de entrada, multiplicados por sus correspondientes pesos, se suman entre ellos:

$$\sum_j (n_{ij}, w_{ij}), \quad \text{donde } j = 1,2,\ldots,n$$

Ilustración 5 – Función Suma de entrada

- Producto: todos los valores de entrada, multiplicados por sus correspondientes pesos, se multiplican entre ellos:

$$\prod_j (n_{ij}, w_{ij}), \quad \text{donde } j = 1,2,\ldots,n$$

Ilustración 6 - Función Producto de entrada

- Máximo: sólo se consideran las entradas más pesadas que son previamente multiplicadas por sus correspondientes pesos:

$$\text{Max}_j (n_{ij}, w_{ij}), \quad \text{donde } j = 1,2,\ldots,n$$

Ilustración 7 - Función Máximo de entrada

2.1.2 Función de activación

La función de activación es la base de las redes de neuronas artificiales. Esta función consiste en obtener el nivel de activación en base a la entrada total a la misma. Entre las funciones de activación más comunes podemos encontrar [15]:

- **Lineal:** el nivel de activación es proporcional a la entrada total. Esta función es inestable debido al incremento desmesurado de sus activaciones. Para solventar esta inestabilidad se emplea la técnica del umbral, a partir del cual las salidas son constantes.

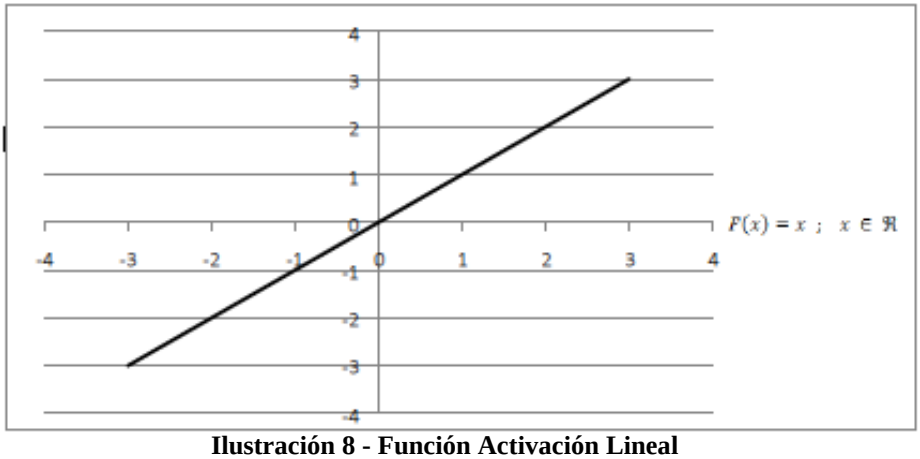

Ilustración 8 - Función Activación Lineal

- **Escalón:** el nivel de activación es un valor discreto comprendido entre el rango (0,1) en función de si la entrada total supera o no un determinado umbral. La desventaja de esta función de activación es que no es derivable en cero, por lo que su proceso de aprendizaje es más limitado.

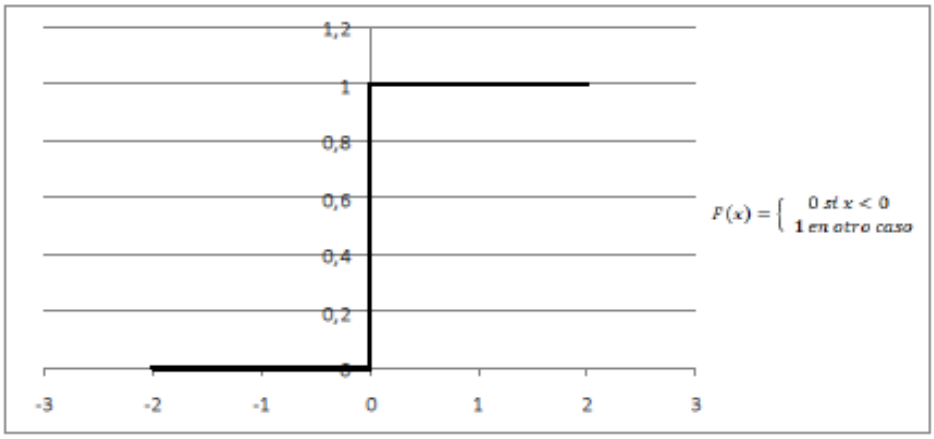

Ilustración 9 - Función Activación Escalón

- **No lineales:** el nivel de activación no es proporcional a la entrada. Esta función se emplea con redes que obtienen una salida continua.

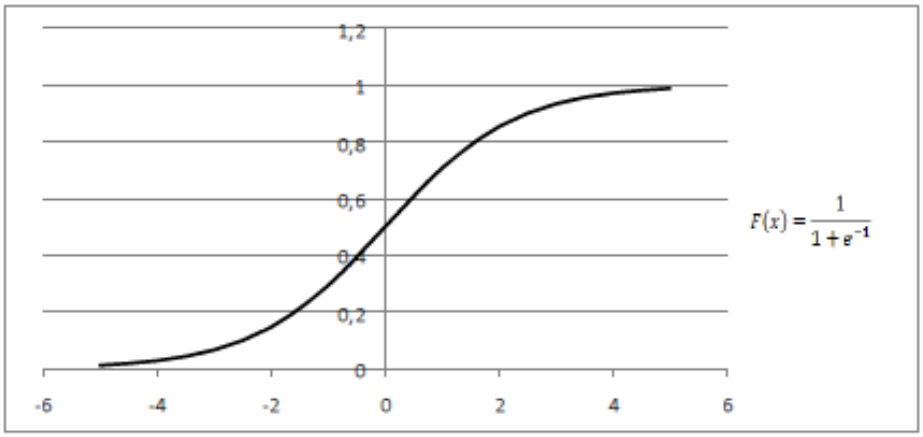

Ilustración 10 - Función Activación no lineal

2.1.3 Función de salida

La función de salida es la encargada de transformar el estado de la neurona en la salida hacia la neurona de la siguiente capa. El resultado se envía a través de la sinapsis. Generalmente, la salida es el propio estado de activación de la neurona.

2.2 Modo de operación

La forma en la que la red neuronal procesa la información que recibe y crea la salida se denomina *modo de operación*.

- **Red estática**: en estas redes, una vez establecidas las entradas, las salidas alcanzan un valor independientemente de las entradas en el instante anterior, y en un tiempo siempre por debajo de una determinada cota. Estas redes tienen una capacidad limitada para sintetizar funciones dependientes del tiempo en comparación con las redes binarias.

- **Red dinámica**: estas redes responden ante las secuencias de entradas, empleando, de manera implícita o explícita, la variable tiempo.

También se pueden dar dos tipos de realimentación:

- Realimentación de la salida, es decir, las salidas de la red son realimentadas a capas anteriores.

- Realimentación del estado, es decir, las salidas producidas en las capas ocultas son realimentadas.

Sin embargo, la realimentación genera en las redes un difícil análisis debido a problemas de convergencia y estabilidad.

2.3 Actualización de los pesos

Las operaciones que se pueden llevar a cabo en una red neuronal para actualizar los pesos de las conexiones y generar una salida son las siguientes:

- **Operación síncrona**: todas las neuronas del sistema generan simultáneamente la salida.

- **Operación asíncrona**: las neuronas del sistema generan aleatoria e independientemente la salida.

2.4 Operaciones de capa

Estas operaciones se producen en cada capa como conjunto, es decir, no en ciertas neuronas que forman la capa, sino en toda ella. Dentro de las operaciones de capa consideramos:

- **Normalización**: todos los elementos de la capa ajustan su salida para obtener un nivel constante de proceso.

- **Competencia**: sólo algunos elementos de la capa ganan y generan una salida.

2.5 Aprendizaje

Como ya se ha mencionado en apartados anteriores, la red neuronal procesa un conjunto de datos o patrones de entrada con el fin de calcular y obtener una salida al problema planteado. El objetivo es obtener un modelo generalizado, es decir, un modelo que sea válido para cualquiera de los patrones que reciba el sistema. Ha de tenerse en cuenta que estos patrones tienen unas características similares ya que son patrones de un mismo problema.

Para poder alcanzar el objetivo de obtener este modelo generalizado, la red neuronal debe pasar por un proceso de aprendizaje, denominado entrenamiento. De la misma manera, el patrón o conjunto de datos de entrada es conocido como patrón de entrenamiento.

Durante la fase de aprendizaje, los únicos cambios que se producen en la red tienen lugar en el valor de los pesos de las conexiones, mientras que la topología de la red, así como cada una de las neuronas de la misma permanecen inalteradas. Debido a esta circunstancia, se puede afirmar que el aprendizaje de una red de neuronas artificial, y por tanto, el modelo final se obtiene mediante la adaptación de los pesos.

El aprendizaje de una red neuronal es un proceso que consiste en hallar y adaptar los valores de los pesos ante unos valores de entrada dados, para que así pueda resolver el problema planteado.

Los pasos a seguir son los siguientes: introducir paulatinamente todos los patrones o ejemplos de aprendizaje, modificar los pesos y una vez que se hayan introducido todos los patrones se comprueba si se cumple cierto criterio de convergencia y si no se cumple, se repite de nuevo este mismo proceso. Los valores de los pesos se pueden modificar después de introducir cada patrón o después de introducirlos todos. El proceso de aprendizaje se finaliza cuando se alcanza un criterio de convergencia, por ejemplo: número fijo de ciclos, error menor que un valor dado o modificación de pesos irrelevante, es decir, los valores de los pesos se estabilizan y no varían.

$$\frac{\partial w_{ij}}{\partial t} = 0$$

Ilustración 11 - Convergencia en el aprendizaje

Pero, ¿cómo se modifican los valores de los pesos? Podemos encontrar tres métodos de aprendizaje dentro de las redes de neuronas artificiales cada uno de los cuales presenta unos criterios diferenciados a la hora de cambiar el valor de las conexiones. [16]:

a. Aprendizaje supervisado
b. Aprendizaje no supervisado
c. Aprendizaje por refuerzo

2.5.1 Aprendizaje supervisado

Este tipo de aprendizaje se caracteriza porque el entrenamiento de la red como su propio nombre indica es supervisado o controlado por un agente externo que proporciona el valor de la salida que se espera recibir de la red ante una entrada determinada. Es decir, se le proporciona a la red el conjunto de datos de entrada y el valor de salida deseado ante esas entradas. En caso de no obtener el resultado esperado, se procederá a modificar el valor de los pesos, con el fin de adaptar estos valores a la salida deseada.

Dentro de este tipo de aprendizaje podemos encontrar dos tipos:

- Aprendizaje por corrección del error: este método consiste en adaptar el valor de los pesos en función del error obtenido en la salida. Algunos algoritmos de este tipo son: el Perceptron, aprendizaje Delta, también conocido como regla del mínimo error cuadrado (LMS Error: Least Mean Squared Error) o el Backpropagation o LMS multicapa

- Aprendizaje estocástico: este método consiste en modificar aleatoriamente los valores de los pesos y evaluar, en función de la salida deseada, su efecto.

2.5.2 Aprendizaje no supervisado

Este tipo de aprendizaje, también denominado autosupervisado, no recibe el valor de la salida deseada, es decir, no es controlado ni supervisado por ningún agente externo. Las redes que presentan este tipo de aprendizaje deben ser capaces de encontrar las características y correlaciones necesarias para presentar y generar una salida óptima.

Dentro de este tipo de aprendizaje podemos encontrar dos tipos:

- Aprendizaje hebbiano: este método consiste en extraer las características de los datos de entrada. Las neuronas permitidas en este tipo de aprendizaje son neuronas binarias, es decir, los valores tanto de las entradas como de las salidas son: {-1, 1} o {0, 1} debido a que la regla de aprendizaje de Hebb partía de una neurona que sólo podía tener dos estados, es decir, podía estar activada o desactivada. El fundamento de este aprendizaje es el siguiente "*si dos neuronas Ni y Nj toman el mismo estado simultáneamente (ambas*

activas o ambas inactivas), el peso de la conexión entre ambas se incrementa".

- Aprendizaje competitivo y comparativo: este método consiste en clasificar los patrones de entrada. De esta manera, si la red recibe un patrón de entrada y determina que este patrón pertenece a una clase ya reconocida por la red previamente, se le aplica el modelo calculado para esa clase previamente. Por otro lado, si el patrón recibido es desconocido por la red, se procederá al reajuste de los pesos para clasificar y reconocer esa nueva clase.

2.5.3 Aprendizaje por refuerzo

Este tipo de aprendizaje es una variación del aprendizaje supervisado. En él, hay un supervisor pero éste solamente indica mediante una señal (refuerzo) si la salida obtenida por la red se acerca a la salida esperada. Esta señal de refuerzo puede tomar por tanto dos posibles valores: acierto igual a +1 ó fracaso igual a -1.

2.5.4 Aprendizaje on-line y aprendizaje off-line

Este tipo de aprendizaje representa si durante la fase de aprendizaje, la red es capaz de mantener su funcionamiento habitual (online), o por el contrario, el proceso de aprendizaje supone la desconexión de la misma hasta que el proceso de aprendizaje haya finalizado (offline).

3. Perceptrón Multicapa

En la actualidad, existen multitud de redes neuronales distintas, pero en este proyecto sólo vamos a trabajar con una red de tipo perceptron multicapa backpropagation.

Este tipo de red de neuronas es una generalización del perceptrón simple que puede estar formada por múltiples capas, y nació con el fin de solucionar los problemas que éste tenía. Uno de esos inconvenientes era no poder resolver problemas que no fueran linealmente separables.

El perceptrón multicapa puede ser totalmente o localmente conectado, es decir, todas las neuronas pueden estar conectadas al resto de neuronas de la red, es decir, cada salida de una neurona de la capa "i" es la entrada de todas las neuronas existentes en la siguiente capa "i+1", o bien pueden estar conectadas sólo a algunas de ellas, es decir, cada neurona de la capa "i" es la entrada de algunas de las neuronas de la siguiente capa "i+1".

Tras numerosos análisis se ha demostrado que este tipo de red de neuronas es un aproximador universal, esto quiere decir, que este tipo de red de neuronas puede aproximar relaciones no lineales entre los datos de entrada y salida.

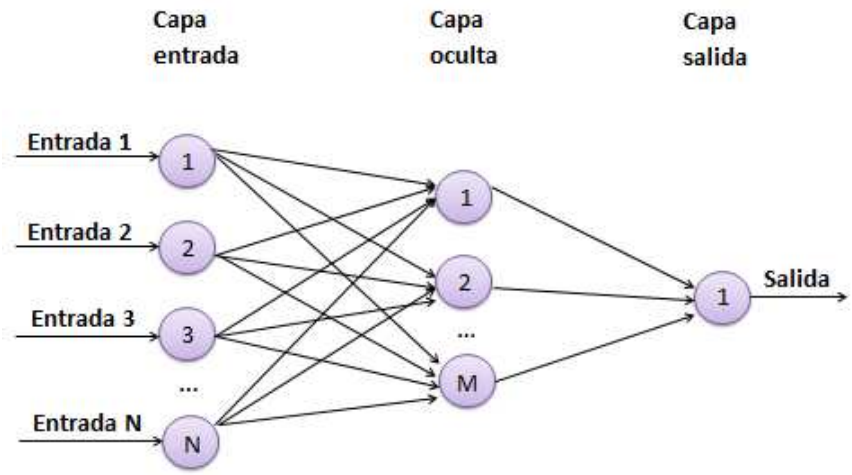

Ilustración 12 – Perceptrón

Sin embargo, en esta red no son todo ventajas. Algunos de los inconvenientes que presenta esta red son:

- Mala extrapolación: si el entrenamiento de la red es malo o insuficiente, los resultados obtenidos en las salidas pueden ser imprecisos.
- Función de error: en esta función existen unos mínimos locales que dificultan el entrenamiento de la red, ya que una vez que se ha alcanzado este mínimo el entrenamiento finaliza a pesar de que no se haya obtenido la convergencia establecida.

Cuando se dan estas malas situaciones se debe considerar la idea de modificar la topología de la red, modificar los pesos al comenzar el entrenamiento, modificar de alguna manera, ya sea el orden o el contenido, el conjunto de entrenamiento, etc.

El algoritmo conocido como *propagación hacia atrás* , retropropagación del error o regla delta generalizada, es un algoritmo utilizado para entrenar este tipo de redes multicapa, por ello, a menudo el perceptrón multicapa es conocido como algoritmo de retropropagación.

3.1 Algoritmo de propagación hacia atrás o Backpropagation

Como se indicó en el Aprendizaje supervisado, el algoritmo Backpropagation presenta un aprendizaje supervisado que aplica un proceso de propagación en dos fases. Por un lado, tras aplicar un patrón a la entrada de nuestra red, lo propagamos por todas las siguientes capas de la red hasta obtener una salida. Esta señal de salida obtenida es comparada con la salida que esperábamos obtener, y a partir de ahí calculamos una señal de error por cada salida obtenida. [17].

Por otro lado, tras haber obtenido la señal o señales de error, las propagamos hacía atrás, es decir, realizamos el camino inverso desde la capa de salida hasta llegar a la capa de entrada, pasando por todas las neuronas de las capas ocultas en el supuesto caso de que haya más de una capa oculta. Sin embargo, en este proceso de propagación hacia atrás, las neuronas no reciben la señal total del error sino una parte de ésta, ya que cada neurona ha influido en la señal en una parte de la misma, es decir, todas las neuronas de la red reciben una señal de error que describe su contribución relativa al

error total. Este proceso se repite, capa por capa, hasta que todas las neuronas hayan recibido la parte de la señal que les corresponde.

Una vez que se realizan las dos fases de propagación (hacia delante y hacia atrás), se produce una actualización de los pesos de las conexiones en función de la señal de error percibida, con el fin de que la red converja.

Gracias a este proceso, a medida que la red es entrenada, las capas ocultas se van organizando por sí mismas de manera que son capaces, a medida que avanza el aprendizaje, de reconocer características de la entrada.

Una vez que ha finalizado el entrenamiento de la red, siempre que el patrón de entrada siga las características que dichas neuronas han aprendido durante su fase de entrenamiento, estás neuronas serán capaces de responder con una salida activa ante un patrón de entrada dado, aunque este esté incompleto o contenga ruido.

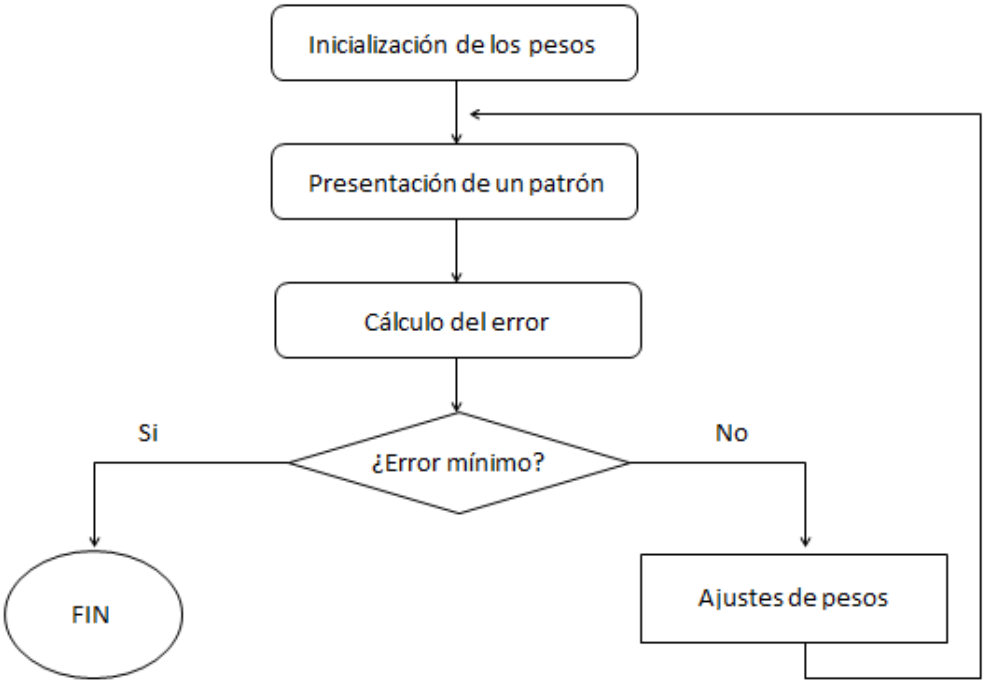

Ilustración 13 - Algoritmo Perceptrón

3.1.1 Descripción del Algoritmo

Una red de neuronas de Backpropagation realiza como ya hemos mencionado bajo un aprendizaje supervisado, por lo que necesita un patrón de entrenamiento donde también vaya indicada la salida que se espera de esas entradas. Este algoritmo basa su tarea de actualización de pesos en base al error medio cuadrático. El patrón de entrenamiento presenta el siguiente formato:

$$\{p_1,q_1\},\{p_2,q_2\},\ldots,\{p_i,q_i\}$$

Donde p_i es una entrada a la red y q_i es la correspondiente salida deseada para el patrón i-ésimo.

La estructura de la red será creada por el diseñador, y constará de pi neuronas de entrada, es decir, una neurona por cada dato de entrada; una o varias capas ocultas con sus correspondientes neuronas; y q neuronas de salida. Cabe destacar que tanto el número de capas ocultas como el número de neuronas que la forman dependen del diseñador, ya que no hay ninguna técnica que determine el número óptimo. Por tanto, la elección del número de capas y del número de neuronas que la forman se basará en la experiencia.

Una vez que se tiene diseñada la red se inicializan todos los parámetros.

A continuación se van a detallar todas las operaciones que tienen lugar para aplicar el algoritmo:

1. El primer paso que hay que realizar se divide a su vez en dos tareas:
   - Inicialización aleatoria de los pesos de cada una de las conexiones de la red con valores pequeños.
   - Proporcionar a la red un modelo de entrada, así como indicar la salida deseada que vamos a esperar de la red para ese patrón de entrada.

2. A continuación, calculamos la salida de la red tras pasar por todas las capas ocultas de la red, a partir de los datos de entrada proporcionados. El procedimiento a seguir es el siguiente:
   - Primero, calculamos el valor de las entradas netas para cada una de las neuronas ocultas a partir de las neuronas de entrada. Para una neurona *j* oculta, el valor será:

$$net_{pj}^{h} = \sum_{i=1}^{N} w_{ji}^{h} x_{pi} + \theta_{j}^{h}$$

Ilustración 14 – Cálculo de las entradas

Donde:

- *h* representa las magnitudes de la capa oculta
- *p* representa al vector *p* de entrenamiento
- *j* es la neurona oculta.
- θ actúa como una entrada más, y es opcional.

   - Una vez realizado este primer paso, se obtienen las salidas de cada una de las neuronas ocultas:

$$y_{pj} = f_{j}^{h}(net_{pj}^{h})$$

Ilustración 15 – Cálculo de las salidas de las capas ocultas

Tras estos dos pasos, realizamos el mismo procedimiento para calcular la salida de la o las neuronas de salida:

$$net_{pk}^{0} = \sum_{j=1}^{L} w_{kj}^{0}\, x_{pj} + \theta_{k}^{0}$$

$$y_{pk} = f_{k}^{0}(net_{pk}^{0})$$

Ilustración 16 – Cálculo de la salida de la red

3. Tras obtener la salida de la red, la comparamos con la salida esperada y calculamos los términos de error para cada neurona:

   - Si *k* es la neurona de la capa de salida, el valor de δ es:

$$\delta_{pk}^{0} = (d_{pk} - y_{pk})\, f_{k}^{0}(net_{pk}^{0})$$

Ilustración 17 – Cálculo del error de la red

Esta función f debe ser derivable, y puede ser de dos maneras:

- Lineal: $f_k(net_{jk}) = net_{jk}$
- Sigmoidal: $f_k(net_{jk}) = \frac{1}{1+e^{-net_{jk}}}$

Ilustración 18 – Función salida

La función utilizada según como se quiera representar la salida. Por un lado, si queremos salidas binarias utilizaremos la función sigmoidal. En cualquier otro caso, se utilizará la lineal. Por tanto, calculamos las derivadas parciales del error:

- Lineal: $\delta_{pk}^{0} = (d_{pk} - y_{pk})$, donde $f_k^{0\,\prime} = 1$
- Sigmoidal: $\delta_{pk}^{0} = (d_{pk} - y_{pk}) y_{pk} (1 - y_{pk})$, donde $f_k^{0\prime} = f_k^0 (1 - f_k^0) = y_{pk} (1 - y_{pk})$

Ilustración 19 – Cálculo de las derivadas parciales del error de la red

   - Una vez tenemos el error calculado, podemos proceder a calcular la fórmula final donde podemos observar que el error de las capas ocultas de la red es dependiente de todos los términos del error obtenido en la

capa de salida, a partir de lo cual nace el término *backpropagation* o *propagación hacia atrás*. Esta fórmula final es:

$$\delta^{h}_{pk} = f^{h\prime}_{k}(net^{h}_{pj}) \sum_{k} \delta^{0}_{pk} w^{0}_{kj}$$

Ilustración 20 – Fórmula final algoritmo Backpropagation

Nótese, que en caso de que una neurona *j* no sea de salida, el resultado de su derivada parcial tendrá que ser obtenido a partir de valores que ya son conocidos, y otros que si pueden ser evaluados, ya que no podrá ser directamente evaluada al no ser de salida.

1. Ajustamos los pesos de cada una de las neuronas empleando un algoritmo recursivo, comenzado a recorrer la red del final al principio, es decir, hacia atrás desde las neuronas de la capa de salida hacia las de la capa de entrada.

   - Pesos de la capa de salida:

$$\Delta w_{ij} = y_i * y_j$$

$$\Delta w^{0}_{kj}(t+1) = \alpha \delta^{0}_{pk} y_{pj}$$

Ilustración 21 - Cálculo de los pesos de la salida

   - Pesos de las capas ocultas:

$$w^{k}_{ji}(t+1) = w^{k}_{ji}(t) + \Delta w^{k}_{ji}(t+1)$$

$$\Delta w^{k}_{ji}(t+1) = \alpha \delta^{h}_{pj} x_{pi}$$

Ilustración 22 – Cálculo de los pesos de las capas ocultas

2. Por último, tendremos que repetir todo este proceso varias veces cada par de entradas-salidas de prueba que reciba la red. La repetición del proceso terminará cuando obtengamos un término de error lo bastante pequeño para cada uno de los patrones de aprendizaje:

$$E_p = \frac{1}{2} \sum_{k=1}^{M} \delta^{2}_{pk}$$

Ilustración 23 - Error mínimo de la red

Capítulo 5 | Algoritmos genéticos

1. Introducción

En los años 70, John Henry Holland desarrolló los primeros algoritmos evolutivos, dentro de los cuales se encuentran los algoritmos genéticos. [18]

Los Algoritmos Genéticos (AGs) son técnicas adaptativas que ayudan en la resolución de problemas de búsqueda y optimización y que al igual que las redes de neuronas del apartado anterior, están fundamentados en el proceso genético de los organismos vivos. Por tanto, estos algoritmos se fundamentan en teorías como la Teoría de la Evolución de Darwin y estudios de Lamarck, Mendel y Wallace sobre procesos naturales de la evolución biológica y su base genético-molecular.

Lo que los AGs tratan de imitar es la teoría de selección natural y de la supervivencia de los más fuertes postulada por Darwin (1859). En la naturaleza los individuos de una misma población luchan y compiten por los recursos existentes, incluso compiten en la búsqueda de un compañero. Dentro de esta competición, aquellos individuos que tengan un mayor éxito a la hora de sobrevivir y de encontrar un compañero, tendrán una mayor probabilidad de generar más descendencia. Por el contrario, aquellos individuos poco dotados que no ganen en dicha competición, producirán un menor número de descendientes. Ante esta situación, los genes de los individuos más adaptados que logran sobrevivir y procrear se transmitirán en mayor medida a las sucesivas generaciones hacia un mayor número de individuos. Por tanto, si combinamos los genes de varios de los individuos más adaptados, podremos a veces obtener descendientes conocidos como “súper individuos”, cuya adaptación a la naturaleza será mejor que la de sus antecesores. Por tanto, cuanto menor sea la adaptación de un individuo, la probabilidad de que sea seleccionado para reproducirse, es decir, la probabilidad de que sus genes se propaguen en sucesivas generaciones será menor. De esta manera, las especies evolucionan consiguiendo unas características mejores.

Y, ¿cómo imitan este proceso los algoritmos genéticos? Éstos, trabajan con una población o conjunto de individuos cada uno de los cuales representa una posible solución al problema que se trata de resolver. Cada individuo tendrá asociado un valor que equivaldría al grado de efectividad de un individuo para competir en la naturaleza. Cuanto mayor sea este valor, mayor será la probabilidad de que sus genes sean seleccionados para reproducirse con otros genes seleccionados de la misma manera.

De esta manera, se van produciendo sucesivas generaciones que contienen una mayor proporción de mejores genes en comparación con las poblaciones de la generación anterior. Esto se repetirá a lo largo del tiempo de manera sucesiva favoreciendo el cruce de los mejor adaptados y explorando y combinándolos con todo el campo de búsqueda. Si al final, el algoritmo genético logra converger en la solución óptima del problema, significará que este algoritmo ha sido bien diseñado. Es importante conocer que las sucesivas evoluciones del problema hasta alcanzar la solución que converge, es decir, la óptima, depende en gran medida de una buena codificación de las mismas.

Sin embargo, no existe un algoritmo genético que te asegure encontrar la solución óptima al problema, aunque si se puede asegurar que las soluciones que se encuentran si tienen un nivel aceptable en comparación con otros algoritmos o técnicas no especializadas que se emplean en estas circunstancias. La aplicación de los AGs se relaciona con aquellos problemas que no cuentan con técnicas especializadas para su resolución; incluso, aunque estos problemas tengan dichas técnicas, y éstas funcionen bien, pueden sufrir mejoras combinándolas con algoritmos genéticos.

El funcionamiento básico de un algoritmo genético queda representado en la siguiente figura:

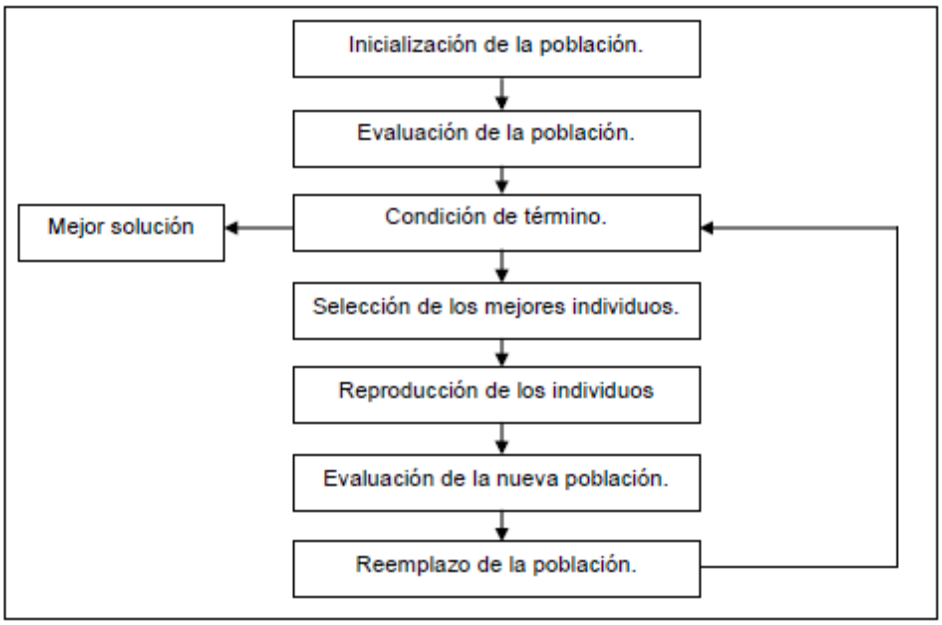

Ilustración 24 - Pasos de un Algoritmo Genético

2. Codificación

El algoritmo genético simple, o también conocido como Canónico está representado en la siguiente figura:

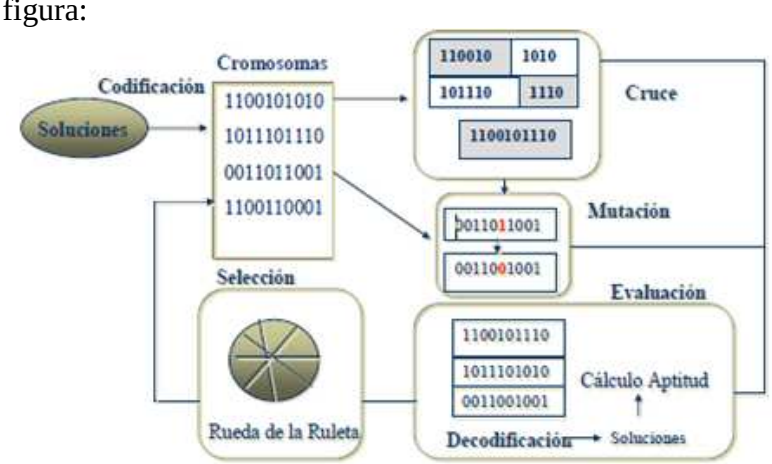

Ilustración 25 – Esquema Algoritmo genético simple

Como ya hemos mencionado anteriormente, es muy importante tener una buena codificación, es decir, que el problema esté bien representado para de esta manera poder obtener el mejor resultado posible. Además de la codificación representación de los datos, se requiere una *función de fitness* o adaptación al problema, la cual asigna un valor a cada posible solución codificada.

Se supone que las posibles soluciones del problema, es decir, los individuos, pueden representarse como un conjunto de parámetros que serán los denominados *genes*, los cuales no tienen por qué representarse por el alfabeto {0,1}, aunque buena parte de la teoría que fundamenta los algoritmos genéticos emplea dicho alfabeto. El conjunto o ristra de valores que forman todos los genes es conocido como *cromosoma*.

Tanto en términos biológicos como en los algoritmos genéticos se denomina *fenotipo* al conjunto de parámetros que representa un cromosoma en particular. Este fenotipo contiene toda la información necesaria para generar o producir un organismo, que es denominado *genotipo*.

Por tanto, para generar una posible solución al problema y ver cómo de bueno es un genotipo, tendremos que analizar y evaluar el fenotipo que puede ser computado a partir del cromosoma utilizando la función fitness.

La función fitness es la parte más crítica de un algoritmo genético. Dado un cromosoma concreto, la función fitness dará como resultado un valor que se le asignará al individuo, y este valor reflejará el nivel de adaptación del individuo al problema y ayudará a determinar la evolución de la población. Esta función fitness debe ser diseñada exclusivamente para cada problema.

Durante toda la fase de reproducción, se seleccionan los individuos de la población para cruzarse y dar lugar a nuevos descendientes. Los padres se seleccionan al azar empleando un proceso que sea favorable a los individuos que estén mejor adaptados, es decir, favoreciendo la evolución de los mejor adaptados. Este proceso, denominado ruleta sesgada, es posible ya que cada individuo lleva asociada una probabilidad de ser seleccionado, y esta probabilidad es proporcional a su función de fitness. Por tanto, los individuos mejor adaptados es decir, los que tengan una mayor probabilidad se escogerán varias veces por generación, mientras que los que tienen baja probabilidad se escogerán menos veces.

Una vez que tenemos seleccionados a los padres, combinamos sus cromosomas correspondientes realizando dos operaciones: *cruce* y *mutación*.

- **Cruce:**

Lo que esta operación realiza es coger la ristra de los cromosomas de los dos padres y cortarla en una posición aleatoria para dar lugar a dos subristras, dos iniciales y dos finales. A partir de este punto se intercambiaran las ristras finales dando lugar a dos nuevos cromosomas formados por genes de los dos padres. Normalmente, la operación de cruce no se realiza en todos los individuos que van a ser emparejados, sino que esta operación se realiza aleatoriamente bajo una cierta probabilidad. Dicha probabilidad se encuentra normalmente comprendida entre 0.5 y 1.0. Si no se produce esta operación, la descendencia se obtiene duplicando los padres.

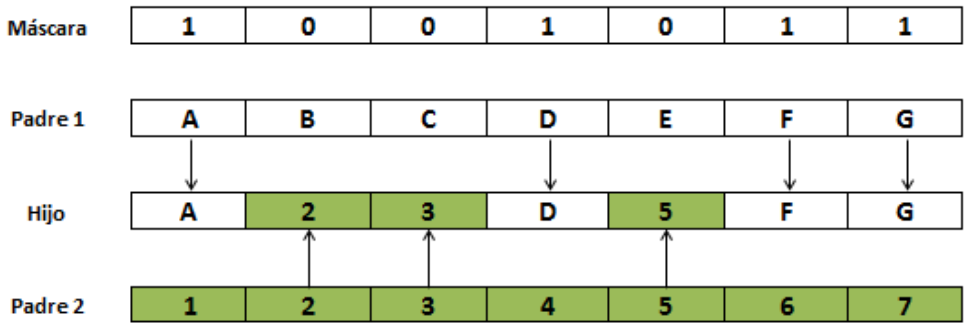

Ilustración 26 - Cruce AG

- **Mutación:**

Lo que esta operación realiza es alterar de manera aleatoria (habitualmente con una probabilidad baja) cada gen que compone el cromosoma. Esta operación se realiza a cada descendiente de manera individual. Lo que se consigue con este proceso es asegurar que todos los puntos del área de búsqueda tienen alguna probabilidad de ser examinados y seleccionados. Es la manera de asegurar la convergencia de los algoritmos genéticos.

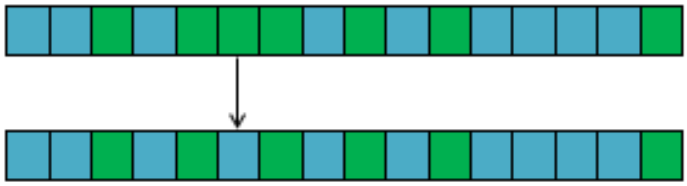

Ilustración 27 - Mutación

Es muy útil conocer con exactitud la definición de convergencia que introdujo De Jong (1975) [19] en este campo, incluida en su tesis doctoral. Si el Algoritmo Genético ha sido correctamente implementado, la población evolucionará a lo largo de las generaciones sucesivas de tal manera que la adaptación media extendida a todos los individuos de la población, así como la adaptación del mejor individuo se irán incrementando hacia el óptimo global.

El concepto de convergencia está relacionado con la progresión hacia la uniformidad: un gen ha convergido cuando al menos el 95 % de los individuos de la población comparten el mismo valor para dicho gen. Se dice que la población converge cuando todos los genes han convergido. Se puede generalizar dicha definición al caso en que al menos un β% de los individuos de la población hayan convergido.

Capítulo 6 | Metodología de desarrollo

1. Introducción

Durante todo el proyecto estamos trabajando con equipos de trabajo multidisciplinares, por lo que la metodología de desarrollo de software que vamos a seguir es una metodología ágil. Las metodologías ágiles de desarrollo de software no son más que métodos de ingeniería de software que se basan en el desarrollo incremental e interactivo, es decir, los requisitos del software y las soluciones que se plantean a éstos, van evolucionando gracias a la participación de equipos auto-organizados y multidisciplinares. [20]

El software se desarrolla por iteraciones o unidades de tiempo que deben durar entre una y cuatro semanas. Cada una de las iteraciones que forman el ciclo de desarrollo del software constan de: planificación, análisis de requerimientos, diseño, codificación, revisión y documentación.

Al final de cada iteración lo que se pretende es tener cumplido el objetivo planteado para esa iteración, es decir, tener una solución válida y sin errores de la funcionalidad desarrollada durante la iteración. Al final de cada iteración se plantean nuevos objetivos para la próxima.

Lo que trata de promover los métodos ágiles es la comunicación entre los miembros del equipo cara a cara y no mediante documentación, por lo que para su buen desarrollo los miembros del equipo deben estar localizados en un mismo espacio, o en un espacio cercano que facilite dicha comunicación.

2. Scrum

Dentro de todas las metodologías ágiles que podemos encontrar, en nuestro caso vamos a emplear la metodología Scrum.

Scrum es una metodología de trabajo de gestión y desarrollo de software que como todas las metodologías ágiles se basa en el desarrollo incremental e iterativo del producto. Consiste en definir una serie de roles, los cuales tienen asignadas una serie de prácticas y tareas concretas.

El desarrollo del software se divide en periodos de tiempo denominados sprints. La duración de los sprints suele ser entre una y cuatro semanas, siendo dos y tres semanas lo recomendable. Además, es recomendable que la duración de los sprints sea constante pudiendo variar en una semana de diferencia, y que esta duración sea definida por los miembros del equipo de trabajo en función a su experiencia en otros proyectos.

Al final de cada sprint, deberán haberse alcanzado todos los objetivos planteados para ese periodo de tiempo, siendo posible exponer y presentar los avances realizados al cliente. Estos objetivos, también conocidos como *Sprint Backlog* deben ser fijos durante

el sprint, y sólo podrán ser modificados si el hecho de no modificarlos pone en riesgo el éxito del proyecto.

Los objetivos planteados en cada sprint, se obtienen del *Product Backlog*, que es un documento donde se encuentran ordenados por prioridad los requisitos del producto a realizar. La elección de qué requisitos desarrollar en cada sprint se deciden en una reunión conocida como *Sprint Planning*, en la que el *Product Owner* selecciona los requisitos que van a ser realizados en cada sprint en base a ciertos factores como el tiempo total que se tiene para el desarrollo del producto, dependencias con otros productos, etc, y se los expone a todos los miembros del equipo. De esta manera, los miembros del equipo analizan el trabajo a realizar y determinan qué cantidad de ese trabajo presentado por el Product Owner pueden comprometerse a realizar en el próximo sprint.

Una de las claves más importantes que tiene esta metodología es que tienen en cuenta que durante el desarrollo de un proyecto, los clientes pueden cambiar los requisitos planteados en un principio. La manera en la que Scrum solventa este problema es con los sprints, ya que son periodos cortos de tiempo en los que se desarrolla siempre software útil, y al trabajar en periodos cortos de tiempo es posible reaccionar ante cambios no previstos y responder de manera adecuada.

Aplicar esta metodología tiene una gran ventaja, ya que es muy fácil de aprender para todos los miembros del equipo, y aunque no se haya tenido una experiencia previa a la hora de trabajar con esta metodología requiere poco esfuerzo para comenzar a utilizarla.

2.1 Roles de Scrum

Los principales roles que intervienen son:

a. **Product Owner:** este miembro del equipo desempeña un papel fundamental, ya que representa al cliente dentro del equipo. Es el encargado de asegurar que el equipo trabaja adecuadamente desde el punto de vista del negocio. Además, es el encargado de gestionar el Product Backlog dando prioridad a los requisitos del cliente.

b. **ScrumMaster:** este miembro es el encargado de dejar libres de obstáculos al resto del equipo de manera que no haya ningún tipo de distracción que impida alcanzar el objetivo. No es el líder del equipo, ya que son auto-organizados, pero si es el encargado de que todo el mundo cumpla las reglas, y realice sus tareas asignadas. También interviene en el desarrollo.

c. **Equipo de desarrollo:** son los encargados de entregar el producto. Aunque inicialmente pueden tener tareas diferenciadas, todos pueden colaborar y participar en todo tipo de tareas como diseño, pruebas, documentación, etc.

Otros roles que también intervienen en Scrum pero que no están involucrados directamente con el desarrollo del proyecto son:

d. Stakeholders: son los beneficiarios del producto y los que lo hacen posible, como por ejemplo, los clientes, proveedores, etc. Únicamente colaboran directamente con el equipo en las revisiones del sprint.

e. Managers o administradores: responsables de establecer el ambiente adecuado para desarrollar el proyecto.

2.2 Documentos de Scrum

a. Product Backlog: documento que contiene todos los requisitos deseables para el producto, es decir, representa lo que va a ser desarrollado. Estos requisitos aparecen priorizados dentro del documento.

b. Sprint Backlog: documento donde se indica cómo va a desarrollar el equipo los requisitos establecidos para ese sprint. Las tareas a realizar se estiman en horas, no pudiendo una tarea superar las 16 horas. En cada se superarlas, tendrá que ser subdividida en 2 tareas de menor tiempo.

c. **Burn down:** documento que contiene el burn down chart, que es una gráfica que indica el progreso del proyecto, así como la cantidad de requisitos que no han sido cumplidos al comienzo de cada sprint. La forma óptima de la gráfica es una línea descendente que indica que los requisitos están siendo completados y por lo tanto, han sido bien definidos y organizados en el tiempo. Cuando esta línea llega al eje horizontal indicará el final del proyecto (ningún requisito pendiente). En caso de que surjan nuevos requisitos en mitad del proyecto, la línea reaccionará ascendientemente.

2.3 Reuniones de Scrum

a. Scrum diario o Daily Scrum: se trata de una reunión diaria sobre el estado del proyecto. La reunión cuenta con todos los miembros del equipo, se celebra siempre a la misma hora y en el mismo lugar, y en ella participan todos los miembros del equipo. Cada miembro del equipo debe exponer cada día:

- ¿Qué ha realizado desde el día anterior?
- ¿Qué tiene planificado hacer en el día de hoy?
- ¿Ha surgido algún problema a la hora de trabajar?

b. Scrum de Scrum: se trata de una reunión que se celebra al final de cada sprint y en la que participa un miembro de cada equipo. Esta reunión está enfocada para cuestiones de integración dependencias entre proyectos, etc. Además de las cuestiones anteriores, se plantea:

- ¿Qué ha realizado tu equipo desde la última reunión?
- ¿Qué tiene planificado hacer tu equipo hasta la próxima reunión?
- ¿Hay surgido algún problema o retraso en tu equipo?
- ¿Estás a punto de entregar algo que afecte a otro equipo? ¿Necesitas algo de otro equipo para seguir avanzando?

c. **Planificación del Sprint o Sprint Planning Meeting:** reunión que se lleva a cabo al comienzo de cada sprint, y en la que se deciden los siguientes puntos:
    - ¿Qué trabajo se va a realizar?
    - Preparar, con la participación de todo el equipo, el Sprint Backlog.
    - Acordar la cantidad de ese trabajo que se compromete entregar al final del sprint.

d. **Revisión del Sprint o Sprint Review Meeting:** reunión que se produce al final de cada sprint y en la que se trata lo siguiente:
    - Revisar qué se ha cumplido y qué no.
    - Exponer el trabajo realizado.

e. **Retrospectiva del Sprint o Sprint Retrospective:** reunión que se produce al final de cada sprint y en la que todos los miembros del equipo exponen sus opiniones sobre el sprint recién terminado. El objetivo de esta reunión es proponer mejoras para los siguientes sprints.

PARTE III: ANÁLISIS

Capítulo 7 | Modelo Conceptual

1. Introducción

Este proyecto, *Gestión multi-variable de equipos multi-disciplinarios e internacionales de equipos y proyectos de desarrollo de software*, nace con la intención de beneficiar y facilitar a las empresas la formación de equipos de trabajo. Para ello, proporciona un modelo que permita determinar cuál sería la formación ideal del equipo de trabajo para desarrollar un software determinado, es decir, qué empleados serían los idóneos para desempeñar esas tareas.

Existen infinidad de combinaciones posibles de tipos de diversidad que se pueden dar en los equipos de trabajo. Por ello, surge la necesidad – sobre la cual se basa *Gestión multi-variable de equipos multi-disciplinarios e internacionales de equipos y proyectos de desarrollo de software* – de analizar y procesar todas estas posibles combinaciones, extrayendo información completa y concisa que nos permita generar un patrón sencillo, efectivo, visual y rápido.

Por tanto, desde un punto de vista conceptual y funcional el modelo a desarrollar puede dividirse en tres fases:

- **Análisis de los datos de entrada que recibirá el sistema** a partir de los cuales tendremos que obtener la información necesaria que nos permita conocer el tipo de empleado que cumpliría con el desarrollo de estos requisitos.
- **Gestión de estos datos de entrada a través de un proceso de aprendizaje** a partir del cual ir obteniendo una información gracias a la cual podremos obtener aquel equipo que cumpla las condiciones específicas.
- **Modelo de extracción de información** con el que conseguiremos obtener la formación del equipo de trabajo óptima para el desarrollo del proyecto de software en cuestión.

En este capítulo nos vamos a encargar de analizar los atributos de entrada al sistema y de acotar los valores de los mismos.

2. Análisis de los datos de entrada

En este módulo nos encargaremos de analizar y seleccionar los posibles valores de entrada que podrán tener los factores mencionados en el punto anterior.

Antes de analizar los datos de entrada individualmente, vamos a representar el modelo conceptual del sistema, representando la relación existente entre todos los componentes.

La *Ilustración 28* representa el modelo conceptual del sistema:

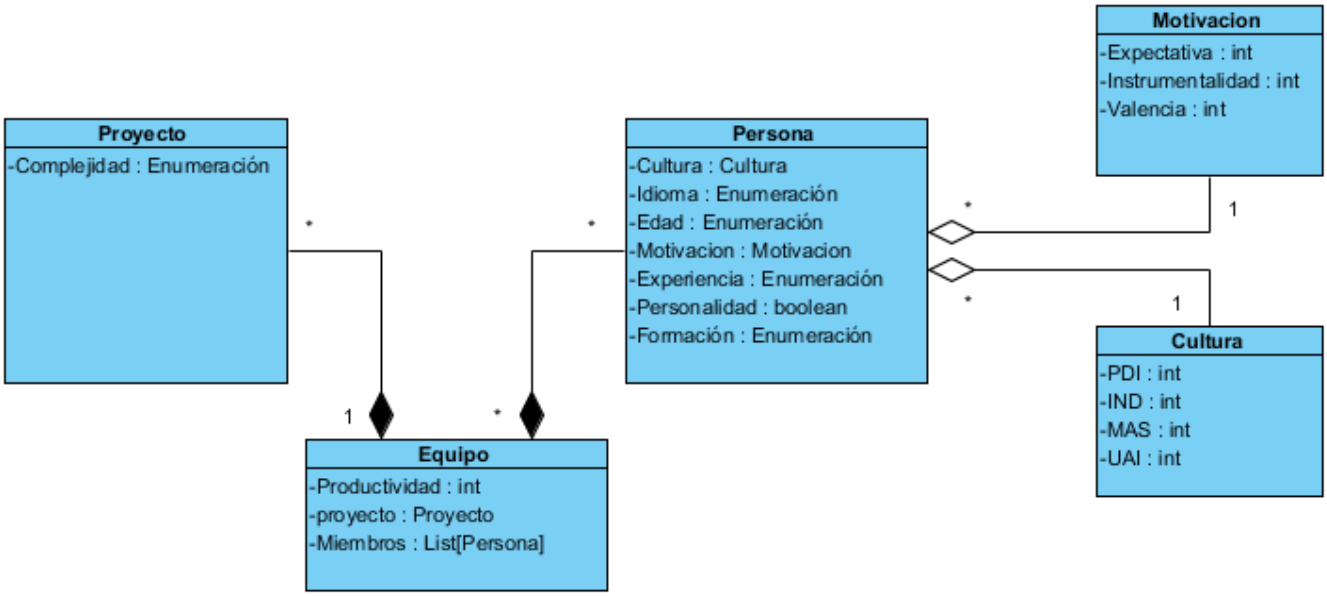

Ilustración 28 - Modelo Conceptual del sistema

A continuación, vamos a identificar cada uno de los conceptos que intervienen en el modelo conceptual.

2.1 Clase Persona

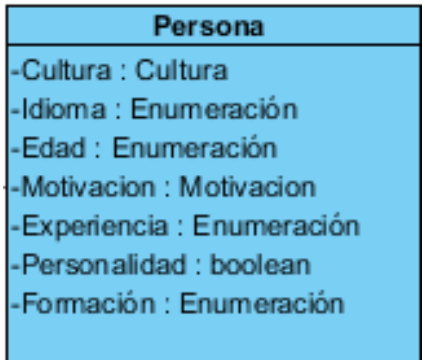

Ilustración 29 - Modelo conceptual persona

Este concepto representa una persona, la cual tendrá una serie de atributos. Cada uno de estos atributos se describe en la siguiente tabla.

Elemento	Significado
Cultura	Representa la cultura de la persona. Es la media de las cuatro características que indican el índice cultural. Una persona sólo puede tener una cultura, pero la misma cultura puede pertenecer a varias personas.
Idioma	Representa el idioma que habla la persona. Puede tomar uno de los siguientes valores: • Inglés • Español • Alemán • Francés/Portugués • Chino
Edad	Representa el grupo de edad al que pertenece la persona. Puede tomar uno de los siguientes valores: • Junior • Semi-senior • Sénior
Motivación	Representa el grado de motivación que tiene la persona para trabajar. Una persona sólo puede tener un grado de motivación, y este grado de motivación es único para cada una.
Experiencia	Representa el grado de experiencia de la persona. Puede tomar uno de los siguientes valores: • Muy alta/Alta • Media • Muy baja/Baja
Personalidad	Representa si la persona posee una personalidad de liderazgo o no.
CapacidadesTecnicas	Representa las capacidades y conocimientos que tiene la persona para desarrollar el proyecto. Puede tomar uno de los siguientes valores: • FP • Otras ingenierías • Técnicas en Informática/Telecomunicaciones • Informática/Telecomunicaciones

Tabla 1 - Modelo conceptual persona

Una vez que identificados cada uno de estos atributos, vamos a explicar y justificar detalladamente cada uno de estos atributos, así como sus valores.

2.1.1 Cultura

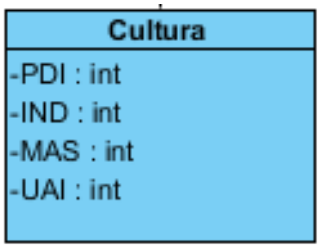

Ilustración 30 - Modelo conceptual cultura

Este concepto representa los tipos de culturas que pueden existir. En la siguiente tabla se detallan cada uno de los elementos que componen la cultura de un país.

Elemento	Significado
PDI	Representa la distancia al poder que tienen los miembros de una cultura.
IDV	Representa el grado de individualismo que tiene la sociedad de una cultura.
MAS	Representa el grado de masculinidad que tiene la sociedad de una cultura.
UAI	Representa el grado de incertidumbre que tiene la sociedad de una cultura.

Tabla 2- Modelo conceptual cultura

Para este estudio, tomaremos como referencia los principales países de desarrollo de software en los diferentes continentes, así como los valores de los índices descritos en el estudio de Hofstede:

País	PDI	IDV	MAS	UAI
Alemania	35	67	66	65
Australia	36	90	61	51
Brasil	69	38	49	76
Canadá	39	80	52	48
China	80	20	66	30
Dinamarca	18	74	16	23

España	57	51	42	86
Estados Unidos	40	91	62	46
India	77	48	56	40
Irlanda	28	70	68	35
México	81	30	69	82
Reino Unido	35	89	66	35
Singapur	74	20	48	8
Suecia	31	71	5	29

Tabla 3 - Índices Hofstede

De la descripción anterior de cada índice y viendo los resultados de la tabla anterior, encontramos que con un valor elevado de los índices IND y UAI, juntos con un valor bajo de PDI y MAS obtendremos unos mejores resultados a la hora de relacionarse. Debido a esto, para poder operar con estos valores y poder crear una teoría a partir de ellos, vamos a invertir el valor de los índices PDI y MAS, de manera que así todos los valores seguirán el mismo patrón: a más valor, mejor.

País	PDI	IDV	MAS	UAI
Alemania	65	67	34	65
Australia	64	90	39	51
Brasil	31	38	51	76
Canadá	61	80	48	48
China	20	20	34	30
Dinamarca	82	74	84	23
España	43	51	58	86
Estados Unidos	60	91	38	46
India	23	48	44	40
Irlanda	72	70	32	35
México	19	30	31	82
Reino Unido	65	89	34	35

Singapur	26	20	52	8
Suecia	69	71	95	29

Tabla 4 - Índices Hofstede ordenados

En la siguiente imagen, podemos comprobar gráficamente las distancias culturales entre tres países elegidos. Podemos comprobar que el área que tienen en común Dinamarca y Brasil es mayor que el área que tienen en común Dinamarca y China, de manera que demuestra que la distancia cultural entre las primeras es menor que entre las segundas.

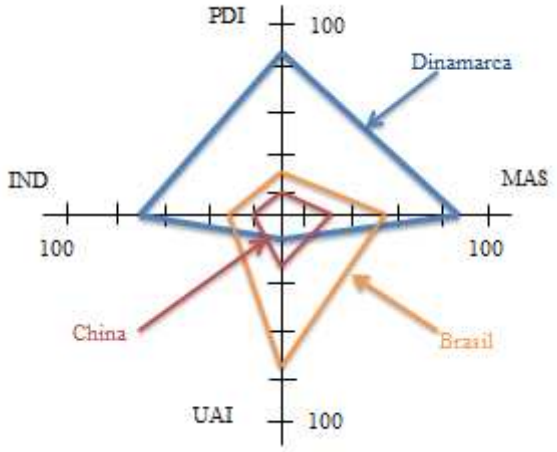

Ilustración 31 - Diferencia cultural según Hofstede

Una vez que tenemos todos los resultados siguiendo el mismo patrón, vamos a extrapolar la siguiente teoría: si la cultura está influenciada por estas cuatro dimensiones, la media de todas ellas dará lugar al valor global de esa cultura, que nos será imprescindible para calcular la distancia cultural (DC) que separa a cada uno de los miembros del equipo.

$$DC = PDIi + INDi + UAIi + MASi / 4$$

País	DC	Valor
Alemania	57.75	6
Australia	61	6
Brasil	49	5
Canadá	59.25	6
China	26	3
Dinamarca	65.75	7
España	59.5	6

Estados Unidos	58.75	6
India	38.75	4
Irlanda	52.25	5
México	40.5	4
Reino Unido	55.75	6
Singapur	26.5	3
Suecia	66	7

Tabla 5 - Diferencias culturales

Todas las culturas estarán englobadas dentro de un rango que irá de 1 a 5, en función del resultado obtenido en el apartado anterior, de manera que cuánto más distancia dentro del rango mayor distancia cultural y por lo tanto mayor dificultad para relacionarse y trabajar en equipo.

1	2	3	4	5
China, Singapur	India, México	Brasil, Irlanda	Australia, Canadá, Alemania, España, UK, USA	Dinamarca, Suecia

Tabla 6 - Grupos culturales

2.1.2 Edad

Como ya hemos mencionado, la diferencia de edad en el equipo de trabajo puede generar conflictos o dificultar la relación entre los miembros del equipo. Por lo tanto, para gestionar las diferentes generaciones en nuestro sistema, vamos a distinguir tres generación o grupos de edad diferentes:

- **Junior:** esta generación abarcará a todos los empleados comprendidos entre las edades de 21 años hasta 29 años.
- **Semi-senior:** esta generación abarcará a todos los empleados comprendidos entre las edades de 30 años hasta 45 años.
- **Sénior:** esta generación abarcará a todos los empleados comprendidos entre las edades de 46 años hasta 65 años.

La manera en la que tratará nuestro sistema a estas 3 generaciones y sus relaciones será la siguiente: tendremos un rango de 1 a 3 que indicará el grado de aportación de cada generación al desarrollo de un proyecto, siendo 1 la menor aportación y 3 la máxima:

Generación	Valor
Junior (J)	1
Sénior (S)	2
Semi-senior (SS)	3

Tabla 7 - Grupos de edad

La razón por la que un semi-senior tiene el máximo valor es fácil, ya que esta generación es la más heterogénea de las dos y en ella podemos encontrar tanto un nivel de experiencia notable así como ideas frescas y nuevas. Por todo esto, podemos observar que el tener en un equipo un empleado semi-senior es equivalente en valor a tener un sénior junto con un junior.

$$SS = S + J$$

2.1.3 Motivación

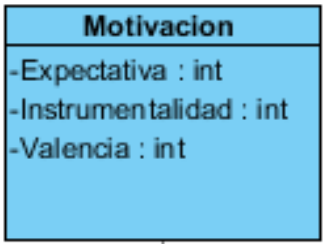

Ilustración 32 - Modelo conceptual motivación

Este concepto representa la motivación para realizar una tarea que tiene cada persona. Cada uno de estos atributos se describe en la siguiente tabla.

Elemento	Significado
Valencia	Representa el nivel de deseo por alcanzar determinada meta.
Instrumentalidad	Representa la probabilidad de recibir una recompensa que estima la persona.
Expectativa	Representa la convicción de una persona en que su trabajo producirá el efecto deseado.

Tabla 8 - Modelo conceptual motivación

Este factor recordemos que estaba compuesto por tres variables según Vroom y que éste sugiere que estos tres factores están relacionados mediante la siguiente ecuación:

Motivación = Expectativas x Instrumentalidad x Valencia

El producto de los tres factores es muy significante, ya que significa que los niveles altos de motivación de cada persona se alcanzarán cuando los tres factores tengan valores positivos y altos. Esto también implica que en el momento que cualquiera de los tres valores sea 0, la motivación siempre será 0, así como si alguno de los resultados es negativo. De manera que, por ejemplo, aunque un empleado crea que su esfuerzo resultará en un buen resultado y que este resultado obtendrá una recompensa, la motivación tendrá valor 0 si la valencia de alcanzar esta recompensa es 0, es decir, si el beneficio o recompensa que recibiría no tuviera valor para él.

2.1.4 Idioma

El tratamiento de este factor se hará generando un rango de valores de 0 a 5 que indicará la importancia del idioma a la hora de trabajar, es decir, el idioma con valor 0 será el que implicará mayor dificultad para comunicarse, mientras que un idioma con valor 5 facilitará mucho más la comunicación debido a que en la actualidad son los idiomas más valoras en el mundo de la ingeniería.

Valor	Idioma
5	Inglés
4	Español
3	Alemán
2	Francés/Portugués
1	Chino
0	Otros

Tabla 9 - Idiomas valorados

2.1.5 Experiencia

Este factor se encargará de medir los niveles de experiencia de cada uno de los miembros del equipo. Vamos a distinguir entre 3 niveles de experiencia que podrán tomar tres posibles valores:

Nivel experiencia	Viabilidad
Muy alto / Alto	2
Medio	1
Muy bajo / Bajo	0

Tabla 10 - Experiencia profesional

2.1.6 Personalidad

En la sección anterior ya hemos comentado que el aspecto fundamental de la personalidad que más nos va a afectar es la capacidad de liderazgo o no que tenga cada miembro del equipo.

Por lo tanto, esta capacidad de liderazgo podrá tomar valor 0 o 1, siendo 1 el indicativo de que ese miembro si tiene una gran capacidad e iniciativa al liderazgo y responsabilidad mientras que 0 indicará una falta de liderazgo y por consiguiente un indicativo de sumisión y seguimiento de las órdenes.

Personalidad	Valor
Líder	1
No líder	0

Tabla 11 - Capacidad liderazgo

2.1.7 Formación y capacidades técnicas

Todos los requisitos mencionados en la sección anterior se adquieren en su mayoría habiendo concluido unos estudios superiores de ingeniería. También se pueden adquirir un subconjunto de estos requisitos cursando otros estudios medios. Por lo tanto, no con todos se adquieren las mismas capacidades, por lo que vamos a hacer distinción entre unos estudios y otros.

Para ello, los estudios podrán tomar valores de entre 0 y 3, siendo 0 el valor mínimo de capacidades que la persona ha adquirido en términos educativos y 3 el máximo valor posible.

Titulación	Viabilidad
Ing. Informática /Telecomunicaciones	3
Ing. Técnicas Informática/Telecomunicaciones	2
Otras (ej.: Industriales)	1
Formación media	0

Tabla 12 – Formación y capacidades técnicas

Nótese que los estudios medios son siempre referidos a estudios relacionados con el desarrollo de software.

2.2 Clase Proyecto

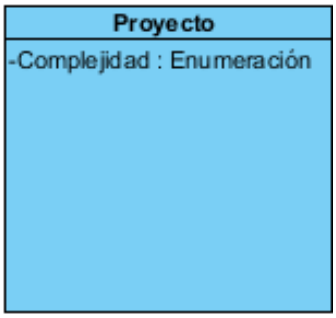

Ilustración 33 - Modelo conceptual proyecto

Este concepto representa el proyecto de software que debe desarrollarse. Un proyecto queda definido por su complejidad y duración. En la siguiente tabla se detalla este atributo.

Elemento	Significado
Complejidad	Representa el grado de complejidad que presenta el proyecto a desarrollar. Puede tomar uno de los siguientes valores: • Corto/Sencillo • Corto/Complejo • Largo/Sencillo • Largo/Complejo

Tabla 13 - Modelo conceptual proyecto

2.2.1 Complejidad y duración del proyecto

Este índice indicará información sobre el proyecto a desarrollar en cuanto a su complejidad y duración. Será un valor entre 0 y 10, donde,

- Los valores comprendidos en el rango [0,5] índica un proyecto de corta duración, y dentro de este rango los valores entre [3,5] indicarán una mayor complejidad dentro de los proyectos a corto plazo.
- Dentro del rango (5,10] representamos los proyectos de larga duración, y como en el rango anterior, dentro del subrango [8,10] encontraremos los proyectos de larga duración más innovadores que tienen una complejidad elevada.

2.3 Clase Equipo

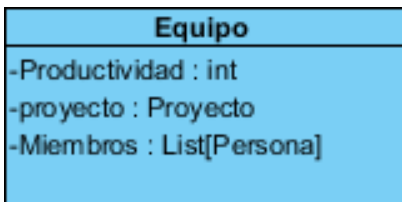

Ilustración 34 - Modelo conceptual equipo

Este concepto representa a un equipo de trabajo. En la siguiente tabla se detallan cada uno de los elementos a tener en cuenta dentro de un equipo de trabajo dentro de nuestro ámbito de estudio.

Elemento	Significado
Productividad	Representa cómo de bueno es el trabajo realizado por el equipo ante un proyecto dado.
Personas	Un equipo puede estar formado por varias personas, y una persona puede pertenecer a distintos equipos.
Proyectos	Un equipo puede participar en varios proyectos, pero un proyecto sólo está asignado a un equipo.

Tabla 14 - Modelo conceptual Equipo

2.3.1 Número de miembros del equipo

Como ya hemos mencionado, este número dependerá de la complejidad del proyecto y de la necesidad de miembros que esta implique, sin embargo, el número máximo de miembros sería 10, de manera que:

Nº Miembros	Viabilidad
3 - 5	Muy Alta / Alta
6 - 10	Media
11 - 15	Muy Baja / Baja

Tabla 15 - Número de miembros del equipo

En función del número de empleados habrá más o menos relaciones interpersonales por lo tanto, las comunicaciones así como el desempeño de tareas se verán afectadas.

La cantidad de vías de comunicación que existen en el equipo depende del nº de empleados y lo calcularemos mediante la fórmula:

$$Nº\ de\ empleados \times \frac{Nº\ de\ empleados - 1}{2}$$

Ilustración 35 - Número de vías de comunicación

Capítulo 8 | Requisitos del sistema

El objetivo de este capítulo es mostrar todos los requisitos para este proyecto, distinguiendo entre requisitos funcionales y no funcionales.

Por requisito funcional entendemos aquel requisito que represente el comportamiento interno del sistema, es decir, cada uno de estos requisitos puede ser identificado como una funcionalidad de nuestro sistema. Los requisitos funcionales son lo que hace nuestro sistema.

Por otro lado, por requisito no funcional entendemos aquel requisito que represente la forma de operar o de funcionar nuestro sistema, es decir, no nos centramos en qué hace sino en cómo lo hace. Dentro de los requisitos no funcionales podemos encontrar de diversos tipos como: seguridad, escalabilidad, mantenimiento, usabilidad, rendimiento, portabilidad, etc.

Cada uno de estos requisitos va a ser definido mediante la siguiente tabla:

Identificador	YY-XX	**Tipo**	
Nombre			
Descripción			
Origen			
Complejidad			

Tabla 16 - Tabla requisitos

Donde:

- **Identificador**: se trata de un identificador unívoco para cada requisito. Este identificador tiene la forma XX-YY, dónde XX indica si es funcional o no funcional pudiendo tomar valores RF o RNF. YY por otro lado, indica el número de requisito que es, y tomará valores desde 1 hasta el número de requisitos de ese tipo haya.

- **Nombre**: es un nombre breve y descriptivo del requisito en cuestión.

- **Descripción**: se trata de una breve descripción del requisito en la que se indicará en que va a consistir.

- **Origen**: este campo representa al sujeto que haya establecido el requisito. Puede tomar los siguientes valores: el cliente, el usuario o el equipo de desarrollo.

- **Complejidad**: este campo índica la dificultad que supone cumplir este requisito. Los posibles valores que puede tomar este campo son: alta, media o baja. Este valor puede variar a medida que avance el desarrollo del proyecto.

1. Requisitos funcionales

En este apartado vamos a enumerar los requisitos funcionales necesarios para el desarrollo de nuestro sistema.

Identificador	RF-01	**Tipo**	Funcional
Nombre	Registrarse		
Descripción	Cada empresa podrá registrarse para poder utilizar el sistema, y así poder a la información que le corresponde. Para ello necesitará el identificador de la empresa y su contraseña.		
Origen	Equipo de desarrollo		
Complejidad	Baja		

Tabla 17 - Requisito funcional 1

Identificador	RF-02	**Tipo**	Funcional
Nombre	Almacenar, obtener, eliminar o editar información de la BBDD		
Descripción	Tanto la información de la empresa y sus empleados, como los resultados obtenidos para cada una, podrán ser almacenados, leídos, eliminados o editados.		
Origen	Equipo de desarrollo		
Complejidad	Media		

Tabla 18 - Requisito funcional 2

Identificador	RF-03	**Tipo**	Funcional
Nombre	Cargar datos de equipos		
Descripción	Cargar los equipos de trabajo que ya se tiene la empresa para calcular su productividad.		
Origen	Equipo de desarrollo		
Complejidad	Baja		

Tabla 19 - Requisito funcional 3

Identificador	RF-04	**Tipo**	Funcional
Nombre	Cargar datos de empleados		
Descripción	Cargar los datos necesarios de cada empleado para el funcionamiento del sistema.		
Origen	Equipo de desarrollo		
Complejidad	Baja		

Tabla 20 - Requisito funcional 4

Identificador	RF-05	**Tipo**	Funcional
Nombre	Calcular productividad equipo		
Descripción	Calcular qué equipo, de todos los que ya tiene formados la empresa, obtendrá mayor productividad.		
Origen	Equipo de desarrollo		
Complejidad	Alta		

Tabla 21 - Requisito funcional 5

Identificador	RF-06	**Tipo**	Funcional
Nombre	Calcular mejor equipo para un proyecto		
Descripción	Calcular la formación del equipo que genera mayor productividad para un proyecto.		
Origen	Equipo de desarrollo		
Complejidad	Alta		

Tabla 22 - Requisito funcional 6

Identificador	RF-07	**Tipo**	Funcional
Nombre	Razonar resultados productividad obtenidos		
Descripción	Una vez calculada la productividad, gestionaremos los datos obtenidos para que sean más comprensibles para el usuario.		
Origen	Equipo de desarrollo		
Complejidad	Media		

Tabla 23 - Requisito funcional 7

Identificador	RF-08	**Tipo**	Funcional
Nombre	Calcular el coste del equipo		
Descripción	Una vez obtenido el mejor equipo para un proyecto, podremos calcular su coste para ver si fuera viable.		
Origen	Equipo de desarrollo		
Complejidad	Baja		

Tabla 24 - Requisito funcional 8

2. Requisitos no funcionales

En este apartado vamos a enumerar los requisitos no funcionales que van a ser necesarios para el desarrollo de nuestro sistema.

Identificador	RNF-01	**Tipo**	No Funcional
Nombre	Seguridad del sistema		
Descripción	Toda la información que intervenga en el sistema, (contraseñas, información de empleados, etc.) deberá ser encriptada y almacenada de manera segura en la BBDD.		
Origen	Equipo de desarrollo		
Complejidad	Alta		

Tabla 25 - Requisito no funcional 1

Identificador	RNF-02	**Tipo**	No Funcional
Nombre	Tiempo de respuesta		
Descripción	El sistema deberá proporcionar respondes al usuario en un periodo de tiempo considerable.		
Origen	Equipo de desarrollo		
Complejidad	Alta		

Tabla 26 - Requisito no funcional 2

Identificador	RNF-03	**Tipo**	No Funcional
Nombre	Legal		
Descripción	El sistema debe cumplir con la ley Orgánica de protección de datos personales.		
Origen	Equipo de desarrollo		
Complejidad	Alta		

Tabla 27 - Requisito no funcional 3

Identificador	RNF-04	**Tipo**	No Funcional
Nombre	Usabilidad		
Descripción	El sistema debe proporcionar al usuario una interfaz consistente, amigable, y fácil de usar.		
Origen	Equipo de desarrollo		
Complejidad	Media		

Tabla 28 - Requisito no funcional 4

Identificador	RNF-04	**Tipo**	No Funcional
Nombre	Recuperación del sistema a fallos		
Descripción	El sistema debe ser lo suficientemente robusto para recuperarse después de algún error grave.		
Origen	Equipo de desarrollo		
Complejidad	Alta		

Tabla 29 - Requisito no funcional 5

PART IV: ARCHITECTURE

Capítulo 9 | Arquitectura

3. Introducción

La arquitectura de software según Roger S. Pressman (2003) [21], no es otra cosa que "una descripción de los subsistemas y los componentes de un sistema informático y las relaciones entre ellos". Los aspectos estáticos y dinámicos del software que vamos a desarrollar es lo que se trata de representar en una arquitectura de este tipo.

Existen distintos tipos de arquitecturas del software como por ejemplo las arquitecturas en capas o la arquitectura MVC (Modelo-Vista-Controlador). La arquitectura seleccionada se escoge en la fase de elaboración del producto, de manera que podremos comprender de manera más sencilla el funcionamiento del sistema, organizar el desarrollo de los componentes que lo forman, etc.

4. Diseño de la arquitectura del sistema

En nuestro caso, el patrón seleccionado es una arquitectura por capas. Este tipo de arquitectura puede estar formado desde dos capas hasta N capas o niveles. La principal característica de este tipo de patrón es que el desarrollo del producto software se realiza por módulos, es decir, cada parte del producto se realiza en las distintas capas de la arquitectura seleccionada de manera independiente. Esto implica que en caso de que se produzca algún fallo o se necesite realizar una modificación o actualización, sólo se verá afectada la capa en cuestión, es decir, el resto de capas de la arquitectura no se verán afectadas y por tanto tampoco se verá afectado el correcto funcionamiento de las mismas.

En nuestro caso, la arquitectura estará formada por tres capas. La *Ilustración 36* muestra una visión de la arquitectura.

Ilustración 36 - Arquitectura Software de tres capas

Una arquitectura de tres capas o three-tier es una arquitectura software cliente-servidor y un patrón de diseño de software. La característica principal de este tipo de arquitectura es que las tres capas son módulos independientes entre ellos, lo cual permite que alguna de las tres capas pueda ser modificada o reemplazada sin que las otras capas se vean afectadas. Estas capas pueden estar físicamente distribuidas, lo que implica que los componentes de una capa pueden referenciar o comunicarse únicamente con los componentes de la capa inmediatamente inferior. Este hecho reduce las dependencias entre componentes de manera que la capa más baja no conoce nada de la capa superior o de la interfaz.

Además, este tipo de arquitectura hace a nuestro sistema altamente escalable ya que se pueden ampliar tanto el número de clientes como de servidores en caso de ser necesario, y el control de la aplicación es centralizado por parte del servidor, de manera que evitamos que clientes malintencionados accedan a la información del sistema. Por lo tanto, en el servidor controlamos la integridad y seguridad del sistema.

La arquitectura que hemos considerado para nuestro sistema, posee las siguientes capas:

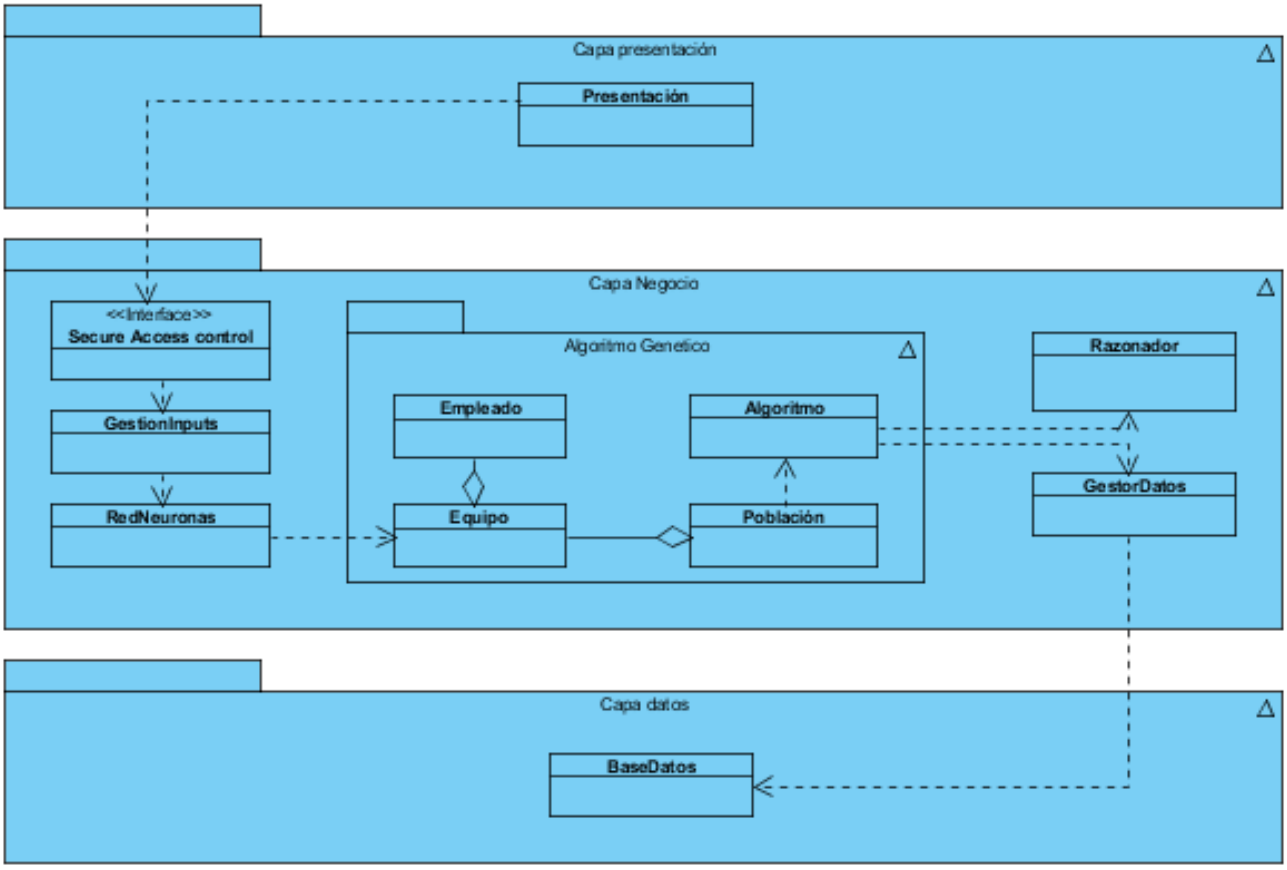

Ilustración 37 - Arquitectura tres capas del sistema

4.1 Capa de presentación

Esta primera capa es el nivel más alto de la aplicación. Esta capa es la que se encarga de interactuar con el usuario y viceversa, es decir, ofrece una visión del sistema al usuario, es decir, en ella se muestra la información de los servicios del sistema al usuario como por ejemplo, introducir los empleados que queremos analizar para formar un equipo. Técnicamente esta capa es conocida como la interfaz gráfica de usuario, también denominada GUI (Graphic User Interface), del sistema que debe ser amigable y usable para el usuario, y que se comunica con la capa de negocio presentándole los datos de salida.

En nuestro sistema, esta capa realiza dos funciones: por un lado, recibe la información que introduce el usuario al sistema, es decir, recibe los datos de entrada, y por otro lado, muestra por pantalla el resultado obtenido por la aplicación para esos datos de entrada.

Esta capa está compuesta por una clase llamada **Presentación** y será la que interactúe con el usuario ofreciéndole una interfaz de usuario para que éste pueda usar el sistema. La composición de esta capa queda reflejada en la *Ilustración 38*:

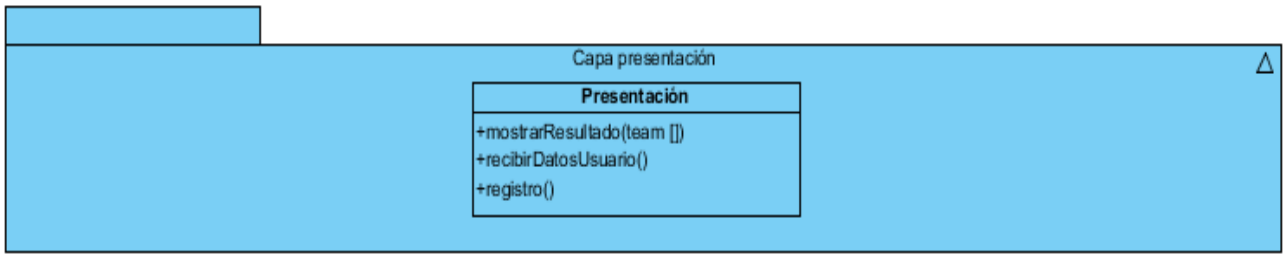

Ilustración 38 - Capa presentación

La funcionalidad que va a ofrecer esta capa es simplemente, por un lado recibir la información introducida por el usuario, y por el otro lado, mostrar al usuario la solución al problema planteado por él.

- **mostrarResultados()**: muestra por pantalla los resultados obtenidos para el equipo introducido por parámetro.
- **RecibirDatos Usuario():** lee los datos introducidos por el usuario al sistema y los envía a la capa de negocio donde el sistema los procesará.

Además, esta clase tendrá una funcionalidad de registro, para que las empresas puedan identificarse y de esta manera puedan acceder a su información interna.

- **registro():** proporciona al usuario la interfaz de registro para entrar al sistema

4.2 Capa de negocio

Esta segunda capa recibe la información introducida por el usuario y recogida en la capa de presentación y con ella se ejecutan las funciones del sistema. Esta capa intermedia representa la lógica de negocio de la aplicación ya que es en esta capa dónde se establecen las reglas del sistema y dónde se controla su funcionalidad.

Esta capa se comunica tanto con la capa de presentación con la que se intercambia información, y con la capa de datos a la que solicita almacenar o recuperar datos ya existentes en el sistema.

Es la capa más compleja del sistema, por lo que para simplificar el desarrollo, esta capa se dividirá en subtareas que representarán la funcionalidad que realizará el sistema. Cada una de estas subtareas representa una de las clases que componen el siguiente diagrama:

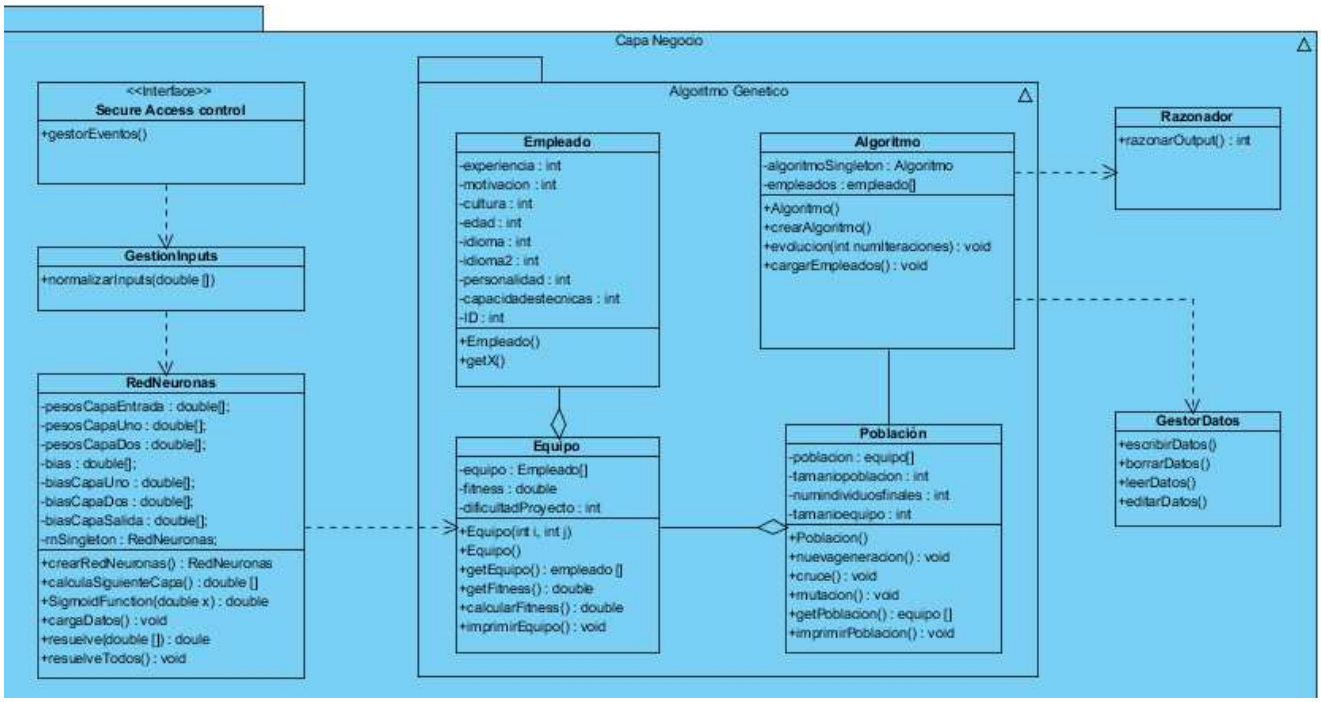

Ilustración 39 - Capa de Negocio

Como se puede observar en la *Ilustración 39*, es una capa en la que intervienen distintas clase por lo que la comunicación entre componentes dificulta la tarea. Debido al tamaño del diagrama, se ha omitido la información de cada una de las clases, la cual va a ser definida a continuación:

4.2.1 La interfaz SecureAccessControl

Esta clase tiene la funcionalidad de actuar como intermediaria entre la capa de presentación y la de negocio. Esta clase recibe toda la información de la capa de presentación y comprueba si el usuario puede realizar o no la tarea que solicita manteniendo la seguridad de la información del sistema.

Ilustración 40 - Interfaz SecureAccessControl

La funcionalidad que va a ofrecer esta clase es descrita a continuación:

- **gestorEventos()**: transfiere el evento que se ha producido en la pantalla para su gestión en la Capa de Negocio en caso de que el usuario pueda realizarla. Si no fuera así mandaría un mensaje de error al usuario.

4.2.2 Clase GestionInputs:

Esta clase se encarga de que la red de neuronas reciba los datos en el formato esperado, ya que sino el comportamiento del sistema no será el adecuado.

Ilustración 41 - Clase gestionInputs

La funcionalidad que va a ofrecer esta clase es descrita a continuación:

- **normalizarInputs ()**: este método comprueba que los datos recibidos por el usuario están normalizados, y en caso de no estarlo los normaliza para enviárselos ya normalizados a la red de neuronas.

4.2.3 Clase Red de Neuronas

Esta clase es la que crea la red de neuronas y generará la función fitness que se empleará en el algoritmo genético.

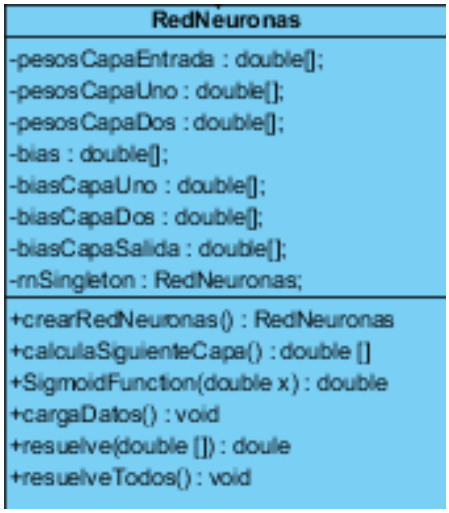

Ilustración 42 - Clase RedNeuronas

La funcionalidad que va a ofrecer esta clase es descrita a continuación:

- **crearRedNeuronas():** este método crea una red de neuronas con todos sus atributos siguiendo el patrón de diseño Singleton. De esta manera, conseguimos garantizar que sólo pueda haber una instancia de la clase RedNeuronas.

- **calculaSiguienteCapa():** este método calcula el valor obtenido por cada neurona de cada capa según la entrada que recibe aplicando la función sigmoidal.

- **sigmoidFunction():** este método realiza la función sigmoidal sobre el valor que recibe y devuelve el resultado de la misma.

- **cargaDatos():** este método carga en sus correspondientes estructuras todos los datos necesarios (bias, pesos, etc) para generar la función fitness de la red neuronal.

- **resuelve():** este método, a partir de los datos de entrada que recibe, da como resultado la productividad del equipo introducido.

- **resuelveTodos():** este método es muy parecido al anterior, solo que en este caso, el método lee de un fichero todos los equipos que quieren ser analizados y da el resultado de la productividad de todos ellos.

4.2.4 Clase algoritmo genético

Esta clase contiene simplemente el método main para la ejecución del algoritmo genético. La funcionalidad que va a ofrecer esta clase por lo tanto, es simplemente obtener el resultado de la productividad que busca el usuario.

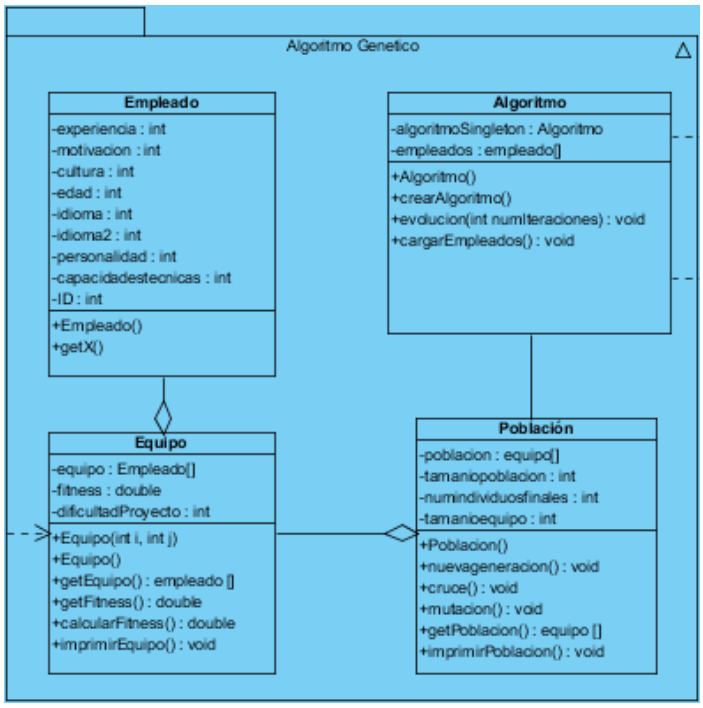

Ilustración 43 - Clase Algoritmo genético

4.2.4.1 Clase Algoritmo

Esta clase es la encargada de realizar la funcionalidad final del algoritmo genético, que no es otra que evolucionar a la población del problema y tratar de obtener el equipo buscado.

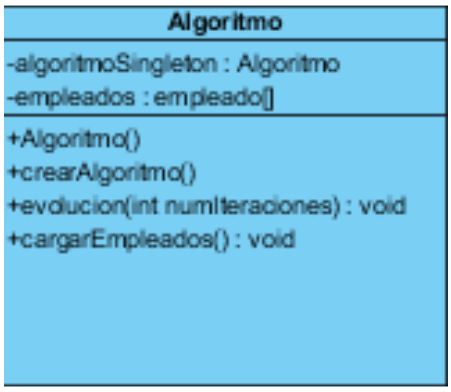

Ilustración 44 - Clase Algoritmo

La funcionalidad que va a ofrecer esta clase es descrita a continuación:

- **algoritmo():** este método es un constructor de la clase que carga los empleados que van a intervenir en la evolución del algoritmo.

- **crearAlgoritmo():** este método se encarga de crear una instancia del tipo Algoritmo, y al igual que la clase redNeuronas, emplea el patrón de diseño Singleton, de tal manera que conseguimos garantizar que sólo pueda haber una instancia de la clase Algoritmo.

- **evolución():** este método se encarga de evolucionar la población tantas veces como reciba por parámetro. Será este método el que nos dé el equipo que estamos buscando.

- **cargarEmpleados():** este método se encarga de cargar en el algoritmo todos los datos de los empleados que van a formar parte de la población que va a intervenir en el algoritmo.

4.2.4.2 Clase empleado

Esta clase es la encargada de definir qué características son las que definen a un empleado, es decir, los atributos o factores que van a ser evaluados por parte de cada empleado para luego poder evaluar la productividad de un equipo.

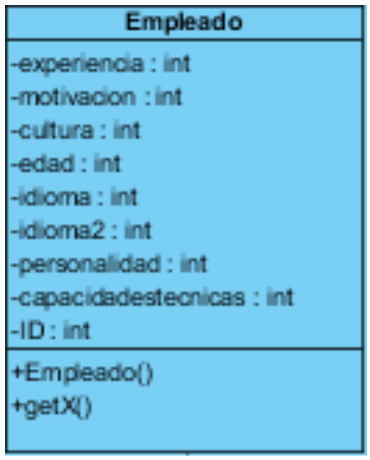

Ilustración 45 - Clase Empleado

La funcionalidad que va a ofrecer esta clase es descrita a continuación:

- **empleado() :** Este método es un constructor de la clase, que lo que crea es un empleado con todas las características que tiene un empleado.

- **getX():** este método permite obtener el atributo de la clase empleado que se desee, pudiendo ser X: *motivación, cultura, edad, idioma, idioma2, personalidad, capacidadestecnicas* o *ID.*

4.2.4.3 Clase Equipo

Esta clase forma los equipos para luego estudiarlos y analizarlos a partir de su productividad generada.

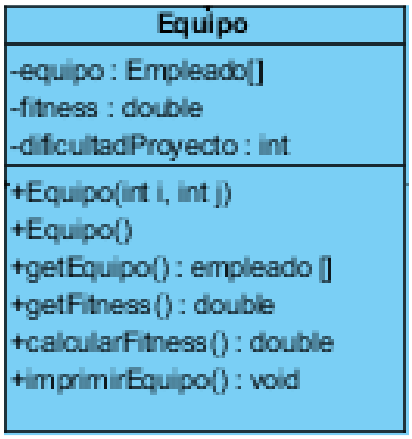

Ilustración 46 - Clase Equipo

La funcionalidad que va a ofrecer esta clase es descrita a continuación:

- **equipo():** este método constructor lo que se encarga es de formar un equipo de trabajo de x personas, siendo x igual a *numeroIntegrantes,* y asociar a este equipo un proyecto y, siendo e igual a *dificultadProyecto*

- **equipo():** método constructor donde ya se le pasa por parámetro la lista de empleados que lo forman.

- **getEquipo():** este método devuelve el valor de la función fitness de un equipo, es decir, la productividad de un equipo concreto.

- **getFitness():**a partir de esta función, obtenemos el fitness de cada equipo o cromosoma usando la función fitness generada a partir de la red de neuronas.

- **calcularFitness():** mediante esta función, calculamos el fitness de un equipo completo.

- **imprimirEquipo():** este método muestra por pantalla los integrantes que forman el equipo.

4.2.4.4 Clase Población

Esta clase trabaja con toda la población de equipos sobre la que se aplicarán las operaciones propias de nuestro algoritmo genético.

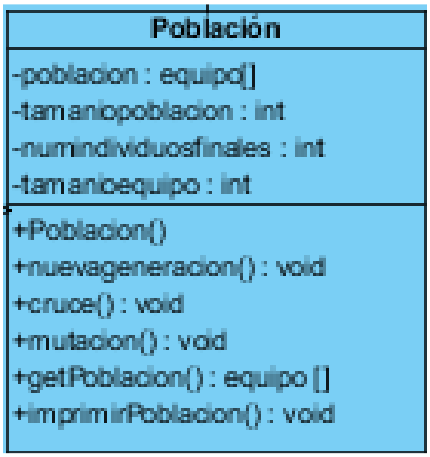

Ilustración 47 - Clase Población

La funcionalidad que va a ofrecer esta clase es descrita a continuación:

- **poblacion():** este método genera la población inicial de equipos (cromosomas).

- **nuevageneracion():**esté método representa la función de selección del algoritmo genético, es decir, selecciona a los padres que van a ser cruzados. En nuestro caso, selecciona a los dos empleados que van a ser cruzados

- **cruce():** esté método representa la función de cruce del algoritmo genético, es decir, esta función realiza el cruzamiento entre los dos empleados escogidos de la población mediante el método anterior.

- **mutacion():** esté método representa la función de mutación del algoritmo genético, es decir, produce una mutación en el empleado seleccionado alterando una de sus características.

- **getPoblacion():** a partir de esta función, obtenemos la población.

- **imprimirPoblacion():** este método muestra por pantalla el conjunto de equipos que forman la población junto con su valor fitness asociado.

4.2.5 Clase GestorDatos

Esta clase es la que se encarga de la comunicación con la Capa de Datos, por lo tanto, esta clase proporcionará a la capa los datos la información que debe escribir, y también, le enviará peticiones de lectura y recibirá la información referente a dicha lectura. En la *Ilustración 48* se muestra la estructura de esta clase.

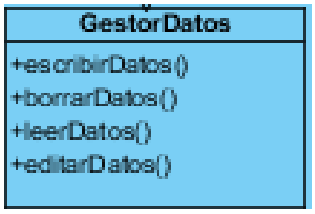

Ilustración 48 - Clase GestorDatos

La funcionalidad que va a ofrecer esta clase es descrita a continuación:

- **escribirDatos():**este método se encarga de escribir y almacenar toda la información del sistema, tanto los datos de la empresa como los resultados obtenidos para cada equipo.
- **borrarDatos():**este método elimina la información que se desea de la base de datos.
- **leerDatos():**este método se encarga de leer de la base de datos la información necesaria para el funcionamiento del sistema, o la información solicitada por el usuario.
- **editarDatos():**este método permite modificar información ya almacenada en la base de datos, como por ejemplo, los datos de la empresa.

4.2.6 La clase Razonador

Esta clase es la que se encarga de, una vez obtenido el resultado de la productividad de un equipo, transformarlo para que el usuario reciba la información de la manera deseada.

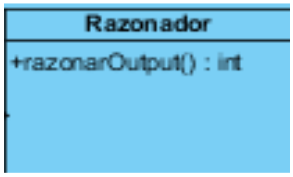

Ilustración 49 - Clase Razonador

La funcionalidad que va a ofrecer esta clase es descrita a continuación:

- **razonarOutput():**este método se encarga de a partir del valor obtenido por el algoritmo genético, dividirlo entre el número de recursos que han participado, es decir, en este caso entre el número de empleados y mostrarle esa información al usuario.

4.3 Capa de datos

Por último, la tercera capa es el nivel más bajo de la aplicación, y consiste en los servidores de bases de datos. Aquí es donde se guarda y almacena la información del sistema. Esta capa mantiene los datos independientes de las dos capas superiores y puede estar formada por uno o varios gestores de bases de datos, mejorando así la escalabilidad y el rendimiento de nuestra aplicación.

En nuestro caso, al ser una aplicación que no es de gran complejidad con un único gestor sería suficiente.

La composición de esta capa consiste, al igual que la capa de presentación en una clase, denominada **BasedeDatos** que también se comunica con la Capa de Negocio, pero que en esta ocasión se encarga de manejar toda la información que se utiliza y genera en la aplicación para cada empresa.

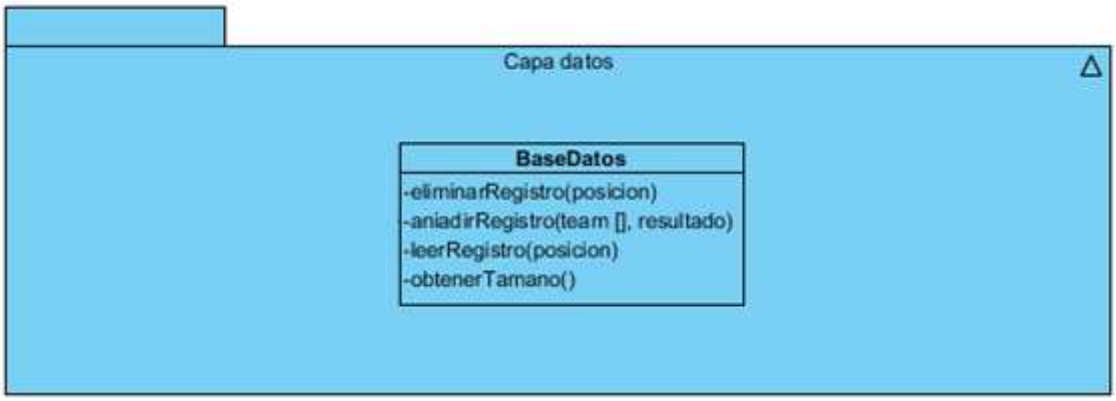

Ilustración 50 - Capa datos

Al igual que en la Capa de Presentación, la funcionalidad que realiza esta capa son operaciones básicas de lectura y escritura en una base de datos.

• **eliminarRegistro()**: borra de la base de datos el registro de la posición introducida por parámetro.

• **añadirRegistro()**: escribe en la base de datos el equipo de trabajo introducido junto con el resultado obtenido para el mismo.

• **leerRegistro()**: devuelve el registro de la posición introducida por parámetro.

• **obtenerTamano()**: devuelve el número de registros que se encuentran almacenados en la base de datos, es decir, el tamaño de la base de datos.

PARTE V: IMPLEMENTACIÓN Y SIMULACIÓN

Capítulo 10 | Esquema general de diseño

En este apartado, vamos a exponer y describir la solución que se ha decidido implementar para desarrollar el sistema. El objetivo principal del sistema es formar equipos de trabajo tratando de obtener el mayor rendimiento del equipo teniendo en cuenta las habilidades, capacidades y características de cada uno de los miembros del equipo y la relación entre ellas.

En este caso hemos optado por evolucionar el algoritmo genético mediante redes neuronales. El genoma del algoritmo genético está formado por los parámetros de la red de neuronas como por ejemplo, el algoritmo de aprendizaje, función de activación, etc.

En este capítulo, expondremos distintas redes neuronales con el fin de obtener la que mejor se adapte al sistema que planteamos. Esta red será usada como función de fitness en nuestro algoritmo genético. Además, también mostraremos en detalle el algoritmo genético implementado.

El orden interno de este capítulo será:

- Solución propuesta
- Diseño y explicación de nuestro algoritmo genético.
- Explicación de los clasificadores diseñados.

1. Solución propuesta

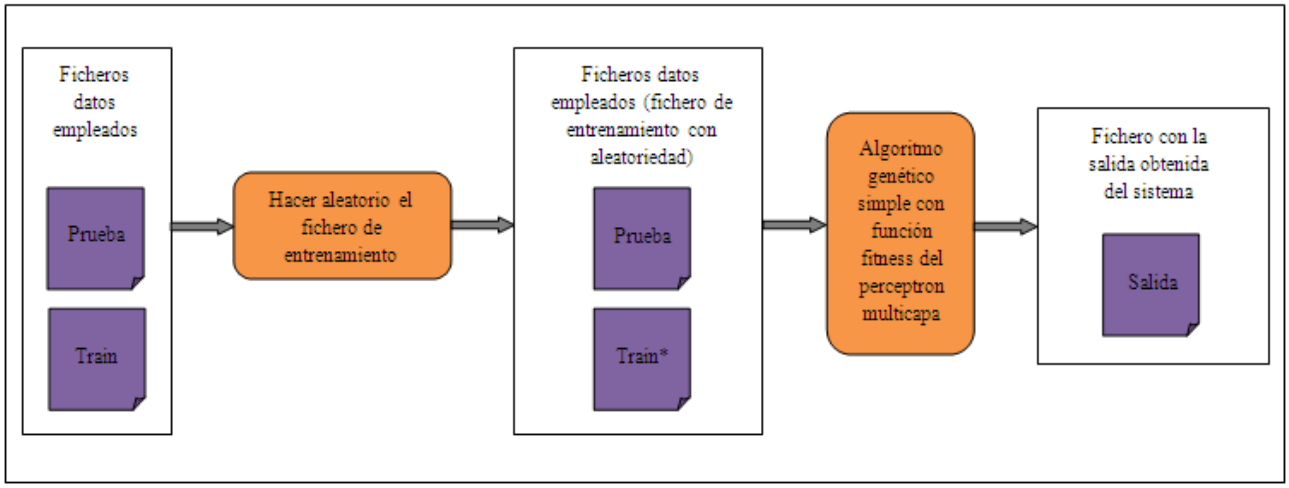

Ilustración 51 - Solución propuesta para el sistema

En el esquema representado en la figura anterior podemos ver el esquema general de la solución propuesta. Como podemos observar en la figura, nuestro sistema contará con un fichero de datos de entrada. A partir de estos datos, e introduciendo dentro del algoritmo la función fitness generada previamente en nuestra red de neuronas, ejecutaremos todos los pasos necesarios para el correcto funcionamiento de nuestro AG.

Al finalizar la ejecución del algoritmo, obtendremos la salida del algoritmo detallada.

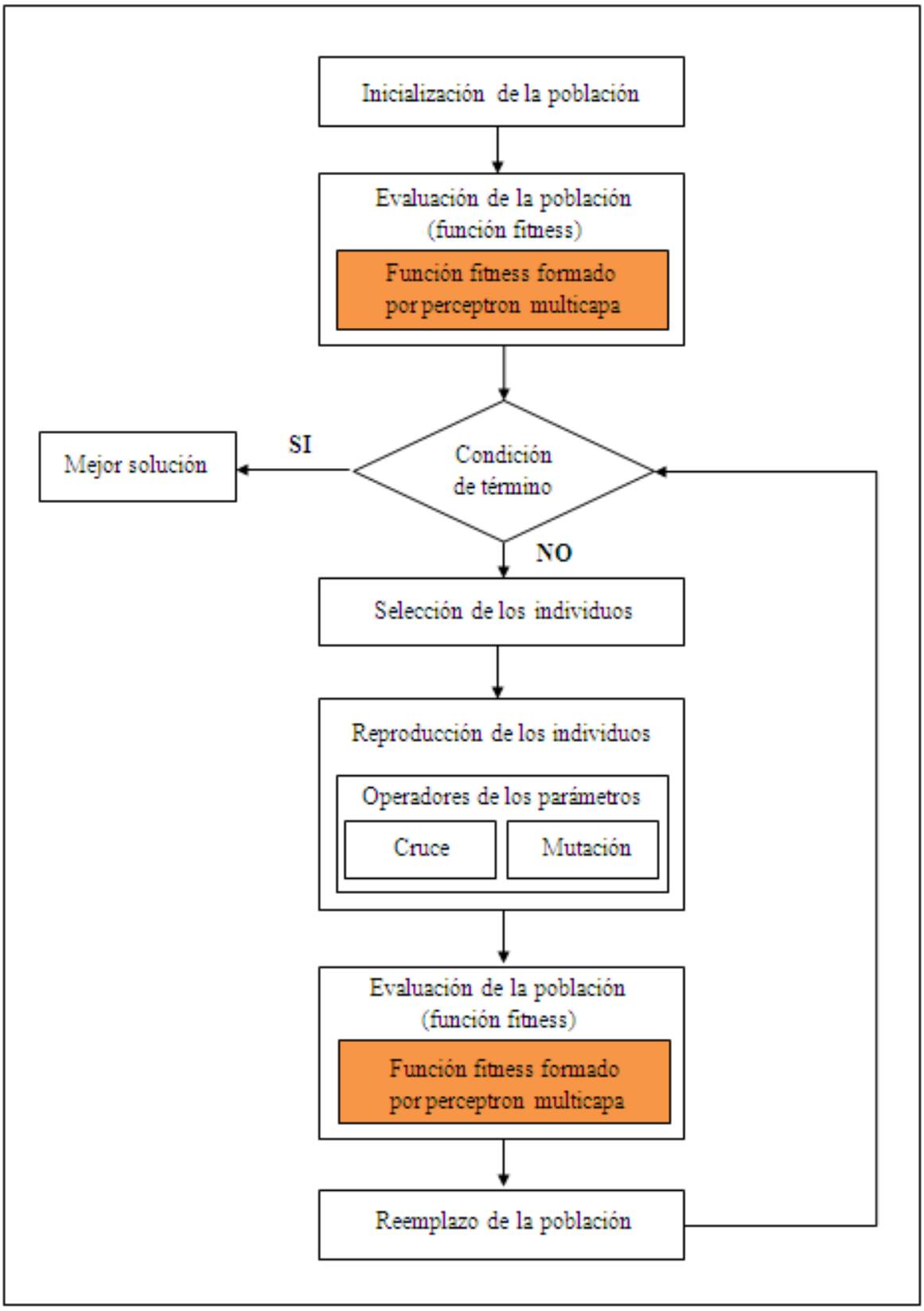

Ilustración 52 - Diagrama de flujo del sistema

Por lo tanto, en resumen, nuestra solución para el problema es un algoritmo genético (Véase Capitulo 5). Además, para la correcta evaluación de los individuos, hemos implementado un clasificador formado por una red de neuronas del tipo perceptron

(Véase Capitulo 4), que emplea el algoritmo backpropagation. Este clasificador constituirá la función de fitness del algoritmo genético.

Como se observa en la *Ilustración 52*, el procedimiento seguido es:

1. Crear una población inicial al azar.
2. Evaluar y evolucionar la población mediante distintos operadores de manera que nos lleve a la mejor solución, reduciendo el número de errores al mínimo posible.

En los siguientes apartados, se procederá a explicar el modelo planteado, comenzando por los clasificadores realizados con un perceptrón multicapa backpropagation, y a continuación explicando el algoritmo genético planteado.

1.1 Justificación de la solución

Como se ha comentado en el apartado anterior, la solución propuesta es una combinación de un algoritmo genético con una red de neuronas, pero ¿por qué no se ha empleado únicamente un algoritmo genético o una red de neuronas?

Dado nuestro problema, una red de neuronas nos serviría para evaluar cómo de buenos son nuestro equipos de trabajo, y qué equipo de los que ya tenemos, sería el mejor para realizar cierto proyecto. Sin embargo, sólo con esta técnica, nuestro sistema se queda cojo, ya que nuestro objetivo es conseguir formar el mejor equipo para generar la mayor productividad no simplemente evaluar la productividad de los equipos que ya tenemos.

De manera que para completar esta función, hemos añadido a nuestro sistema un algoritmo genético que sí será capaz de evaluar cada uno de los individuos de la plantilla de trabajadores individualmente, tratando mediante combinaciones de alcanzar la formación del equipo que dé la productividad deseada.

Nuestro sistema dará como resultado una solución óptima al problema planteado, pero no tiene por qué ser la mejor solución posible, ya que el algoritmo realiza N iteraciones, seleccionando aleatoriamente los miembros del equipo de manera que no siempre tienen porque intervenir los mejores.

La manera de alcanzar la mejor solución sería realizar 2^n iteraciones (en nuestro caso, 2^{200}), siendo n el número de empleados en plantilla, sin embargo, esta solución es muy costosa, por lo que nuestro sistema es una solución alternativa.

1.2 Desarrollo de la red de neuronas aplicada

En este apartado, vamos a proceder a describir de forma más concreta la red de neuronas implementada, justificando la elección del tipo de algoritmo empleado en la red, así como la topología de la red. Además se analizarán las distintas pruebas realizadas con el fin de obtener la función de fitness que mejor se adapte al objetivo final del sistema.

1.2.1 Elección del tipo de red

Como ya hemos mencionado en el Capítulo 5 | Algoritmos genéticoslas redes de neuronas artificiales son sistemas que se caracterizan entre otras cosas por ser capaces de dar solución a problemas que presentan gran conjunto de datos, gracias a su capacidad de aprendizaje. Este hecho es el motivo por el que se decidió hacer uso de las redes de neuronas artificiales para solucionar nuestro problema.

De entre todos los tipos de redes que podíamos haber empleado nos decantamos por el perceptrón multicapa backpropagation. La razón por las que se escogió este tipo de red frente a otras posibilidades que se tenía es que permite resolver problemas que no son linealmente separables como es el caso del perceptrón simple.

Entre los otros tipos de redes que tanteamos, está como hemos mencionado, por un lado, el perceptrón simple, que a pesar de ser menos complejo, genera salidas binarias, lo cual nos proporcionaba un resultado que no nos proporciona suficiente información sobre la salida generada. Además el perceptrón simple sólo es válido para problemas lineales, de manera que para nuestro problema no sería una solución óptima. Por otro lado, tenemos el Adaline cuyas características son similares al perceptrón simple aunque su salida es real y no binaria.

1.2.2 Selección de variables y Preprocesamiento de los datos

Las variables que van a ser empleadas para entrenar la red de neuronas son las ya descritas en apartados anteriores. Sin embargo, cuando se emplea un perceptrón multicapa, se recomienda normalizar los datos [22] [23] [24] [25]. Es decir, todas las variables deben seguir una distribución uniforme de manera que todas tomen sus valores aproximadamente dentro del mismo rango de valores posibles. De esta manera, acotamos el rango de trabajo de la función de activación utilizada en las capas ocultas y de salida.

Teniendo en cuenta este hecho, vamos a normalizar nuestros datos de entrada y salida acotando sus posible valores entre 0 y 1. Para ellos emplearemos la siguiente fórmula:

$$x`^{p} = \frac{x^{p} - \min(x)}{\max(x) - \min(x)}$$

Cabe destacar que todos los datos utilizados para la realización de este sistema han sido proporcionados por Ericsson España. [26]

1.2.3 Elección de los pesos iniciales

Para comenzar a trabajar por primera vez con una red neuronal, debemos establecer los valores iniciales para los pesos, para así poder comenzar a entrenar nuestra red. Estos valores pueden inicializarse de manera aleatoria, sin embargo, es conveniente establecer unas ciertas normas con el fin de minimizar la fase de entrenamiento.

Con el fin de facilitar la fase de entrenamiento, nuestros pesos iniciales serán inicializados de manera aleatoria, pero sus posibles valores estarán acotados al rango -0.5 y 0.5 según la SPSS Inc., (1997) [23]

1.2.4 Tasa de aprendizaje y factor momento

Por un lado, la tasa de aprendizaje es la encargada de controlar el cambio en los valores de los pesos en cada iteración. En 1986, Rumelhart, Hinton y Williams, afirmaron que el valor de la tasa de aprendizaje debe evitar alcanzar valores extremos, y por lo general, suele estar comprendida entre 0.05 y 0.5 [27]

Por otro lado, el factor momento, es el encargado de filtrar las variaciones del error que se generan por la tasa de aprendizaje y además, se encarga de acelerar el proceso de convergencia de los pesos. Según Rumelhart, Hinton y Williams, el valor del factor momento debe ser próximo a 1. [27]

1.2.5 Diferentes arquitecturas planteadas

Para calcular la función fitness que vamos a implementar dentro del algoritmo genético se pueden emplear multitud de variantes del perceptrón multicapa, variando tanto el número de capas ocultas como el número de neuronas dentro de cada capa.

Lo que pretendemos en este apartado es analizar distintas opciones que tenemos e intentar realizar diferentes pruebas con el fin de obtener la que mejor se adapte a nuestro problema. Obviamente, implementar todas las posibles opciones es imposible ya que habría infinitas arquitecturas capaces de resolver el problema, y no existe ninguna técnica capaz de determinar el número óptimo de capas y neuronas ocultas para resolver dicho problema. Por lo tanto, tanto el número de capas ocultas, como el número de neuronas de cada una de estas capas, debe ser elegido por el diseñador.

De entre las pruebas realizadas, a continuación vamos a exponer y explicar las que consideramos que pueden asumir bien el problema, generando un buen resultado. Todas las redes de neuronas implementadas tienen dos capas ocultas, modificando el número de neuronas en cada una de ellas. Las redes implementadas y analizadas son las siguientes:

1.2.5.1 Arquitectura 1

Se tiene un sistema que recibe 5 posibles miembros del equipo que queremos formar, cada uno de ellos con sus atributos y características correspondientes. En este caso estarán caracterizados por 8 atributos. Por lo tanto, nuestra red de neuronas que vamos a crear tendrá:

- En la capa de entrada: 41 neuronas, 8 atributos por cada uno de los cinco miembros, y un último atributo que representa la complejidad del proyecto. Todas las neuronas de entrada se comunican con todas las neuronas de la primera capa intermedia.
- En la capa intermedia, tendrá una capa oculta con 35 neuronas que recibirán la información de todas las neuronas de la capa de entrada, y que una vez procesada esta información se la pasarán a la segunda capa oculta. En esta segunda capa oculta, encontramos 25 neuronas que también reciben la información de todas las neuronas de la primera capa oculta, y que una vez procesada se la transmiten a la capa de salida.
- En la capa de salida: 1 neurona en la capa de salida que nos indicará el valor de la productividad del equipo introducido al sistema.

A continuación vamos a representar gráficamente un ejemplo que ilustre este modelo planteado.

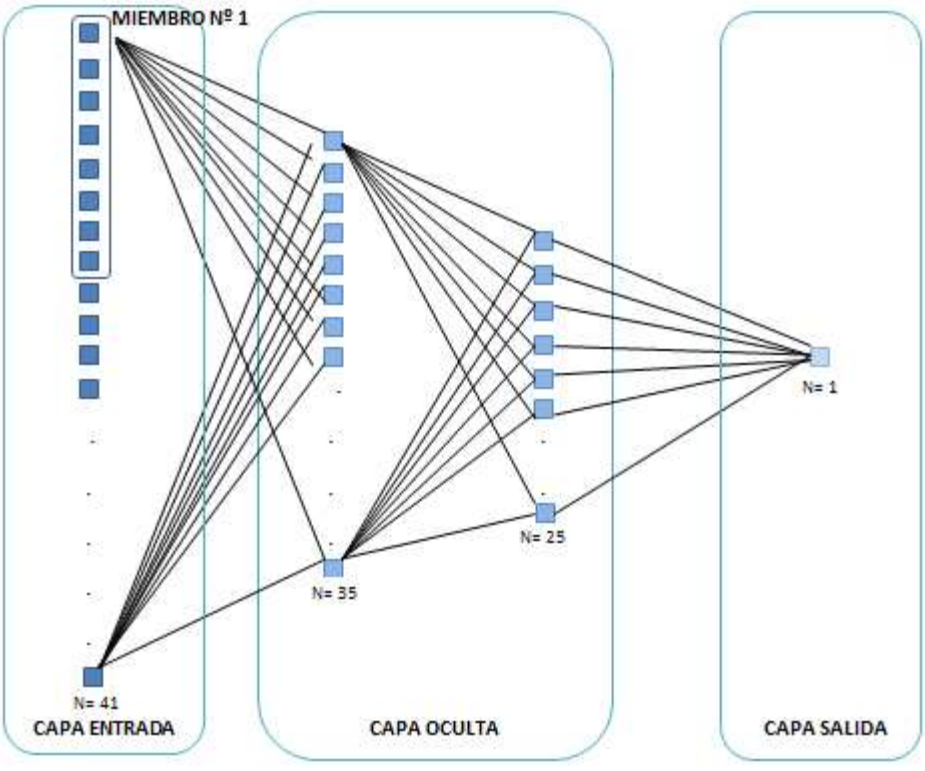

Ilustración 53 – Perceptron multicapa 1

La red de neuronas que hemos creado consta de 2005 conexiones entre sus neuronas debido a que la conexión de la red es total, es decir, todas las neuronas de la red están conectadas entre sí. Por tanto, cada una de estas conexiones tendrá un peso asociado a la misma.

1.2.5.2 Arquitectura 2

Se tiene un sistema que recibe 5 posibles miembros del equipo que queremos formar, cada uno de ellos con sus atributos y características correspondientes. En este caso estarán caracterizados por 8 atributos. Por lo tanto, nuestra red de neuronas que vamos a crear tendrá:

- En la capa de entrada: 41 neuronas, 8 atributos por cada uno de los cinco miembros, y un último atributo que representa la complejidad del proyecto. Todas las neuronas de entrada se comunican con todas las neuronas de la primera capa intermedia.
- En la capa intermedia, tendrá dos capas ocultas. En la primera capa oculta encontraremos 20 neuronas que recibirán la información de todas las neuronas de la capa de entrada, y que una vez procesada esta información se la pasarán a la segunda capa oculta. En esta segunda capa oculta, encontramos 10 neuronas que también reciben la información de todas las neuronas de la primera capa oculta, y que una vez procesada se la transmiten a la capa de salida.
- En la capa de salida: 1 neurona en la capa de salida que nos indicará el valor de la productividad del equipo introducido al sistema.

A continuación vamos a representar gráficamente un ejemplo que ilustre el segundo modelo planteado.

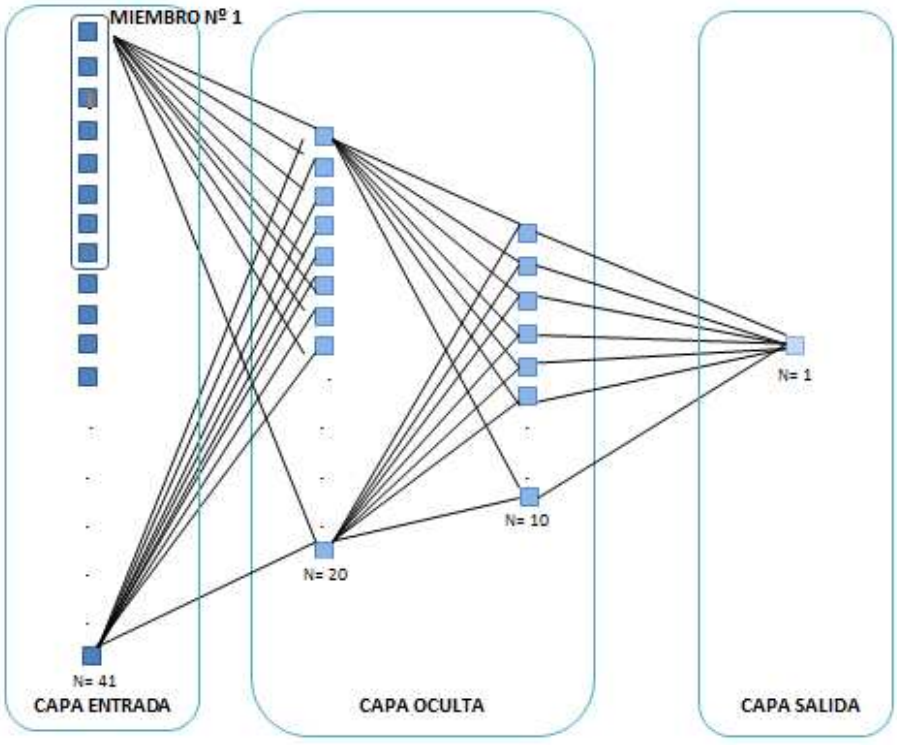

Ilustración 54 - Perceptron multicapa 2

La red de neuronas que hemos creado consta de 1030 conexiones entre sus neuronas debido a que la conexión de la red es total, es decir, todas las neuronas de la red están conectadas entre sí. Por tanto, cada una de estas conexiones tendrá un peso asociado a la misma.

1.2.5.3 Arquitectura 3

Se tiene un sistema que recibe 5 posibles miembros del equipo que queremos formar, cada uno de ellos con sus atributos y características correspondientes. En este caso estarán caracterizados por 8 atributos. Por lo tanto, nuestra red de neuronas que vamos a crear tendrá:

- En la capa de entrada: 41 neuronas, 8 atributos por cada uno de los cinco miembros, y un último atributo que representa la complejidad del proyecto. Todas las neuronas de entrada se comunican con todas las neuronas de la primera capa intermedia.
- En la capa intermedia, tendrá dos capas ocultas. En la primera capa oculta encontraremos 25 neuronas que recibirán la información de todas las neuronas de la capa de entrada, y que una vez procesada esta información se la pasarán a la segunda capa oculta. En esta segunda capa oculta, encontramos 20 neuronas que también reciben la información de todas las neuronas de la primera capa oculta, y que una vez procesada se la transmiten a la capa de salida.
- En la capa de salida: 1 neurona en la capa de salida que nos indicará el valor de la productividad del equipo introducido al sistema.

A continuación vamos a representar gráficamente un ejemplo que ilustre el segundo modelo planteado.

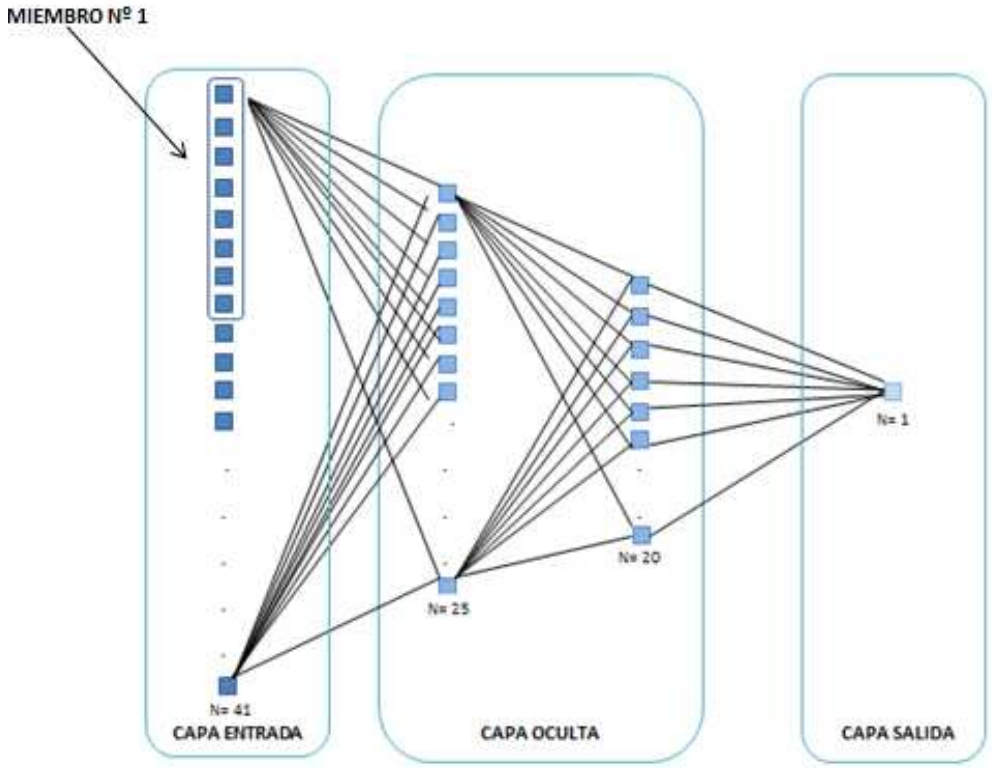

Ilustración 55 - Perceptron multicapa 3

La red de neuronas que hemos creado consta de 1545 conexiones entre sus neuronas debido a que la conexión de la red es total, es decir, todas las neuronas de la red están conectadas entre sí. Por tanto, cada una de estas conexiones tendrá un peso asociado a la misma.

Una vez que nuestra red de neuronas esté entrenada, podremos extraer de ella un modelo que relacione todas las entradas entre sí, y esté modelo será el que utilicemos como función de fitness en nuestro algoritmo genético. Tanto los resultados obtenidos para cada una de las topologías, así como la red elegida, serán descritos en el apartado 2 de este capítulo.

Finalmente habrá que ver el comportamiento del algoritmo genético con el modelo o función obtenido de la red de neuronas.

Las diferentes estructuras de red, así como su fases de funcionamiento se han realizado con JustNNS (Anexo A).

1.3 Descripción del Algoritmo Genético utilizado

En este apartado, vamos a proceder a describir de forma más concreta el algoritmo implementado, entrando en detalle en los métodos y procesos seguidos, así como en detalles técnicos que han sido necesarios considerar para poder llevar a cabo la resolución de nuestro problema.

Más concretamente vamos a explicar la inicialización de los individuos de entrada, la función de evaluación, los métodos de selección, las operaciones realizadas para la generación de nuevos individuos, etc.

Nuestro sistema está focalizado en formar y construir el mejor equipo de trabajo, es decir, diseñar un equipo de trabajo con las mejores habilidades y capacidades globales, no centrándonos en sólo algunas de ellas. Además, el sistema priorizará a aquellos individuos que tengan suficiente conocimiento en todas las áreas o tareas, en vez de aquellos individuos que son expertos en un área o tarea concreta tratando de obtener así un equipo compensado en todos los factores o atributos que afectan al sistema.

El objetivo de nuestro algoritmo genético es proporcionar o generar un conjunto de individuos o un equipo de individuos que a partir de una población inicial va cambiando individuos hasta encontrar el resultado deseado. El esquema de funcionamiento de nuestro algoritmo genético simple o canónico se compone de los siguientes pasos:

a. Generar al azar una población inicial de diferentes cromosomas artificiales.
b. Evaluar a los individuos de la población inicial.
c. Selección de individuos más aptos.
d. Cruce de los individuos seleccionados.
e. Mutación de individuos.

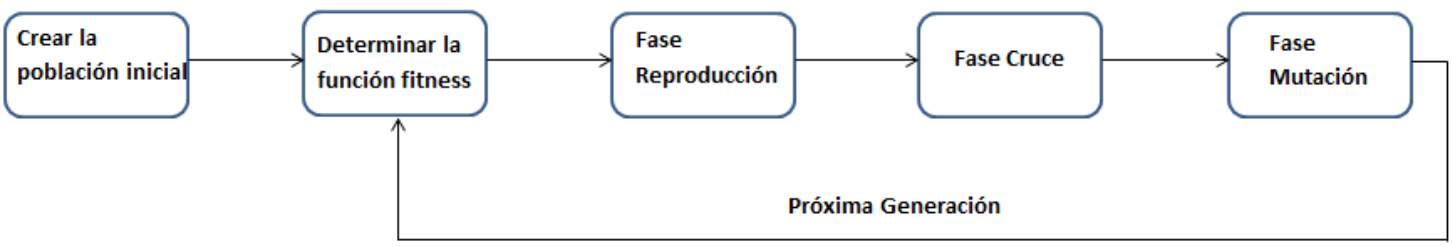

Ilustración 56 - Ciclo de vida de un AG

Los pasos del b al e deberán ser repetidos durante N generaciones con el fin de converger en el resultado deseado.

Como se muestra en la *Ilustración 56*, la estructura de nuestro algoritmo genético sigue el clásico modelo que emplea las dos operaciones de reproducción: cruce y mutación. Todos los pasos que sigue nuestro algoritmo van a ser detallados a continuación:

1.3.1 Generar la población inicial:

El primer paso en un algoritmo genético es crear la población inicial. Para poder generar esta población debemos saber la forma en la que vamos a representar los individuos de la población inicial, es decir, debemos saber la forma en la que vamos a representar los individuos de la población. En términos genéticos, debemos saber la forma de representar los genes de los individuos en el cromosoma. Hay varios tipos de representación, en nuestro caso vamos a utilizar la representación real. En esta representación se utiliza un vector o ristra de valores (cromosoma) cuya longitud es la del número total de genes (o atributos) de cada individuo y el valor que puede tomar cada gen es un número real.

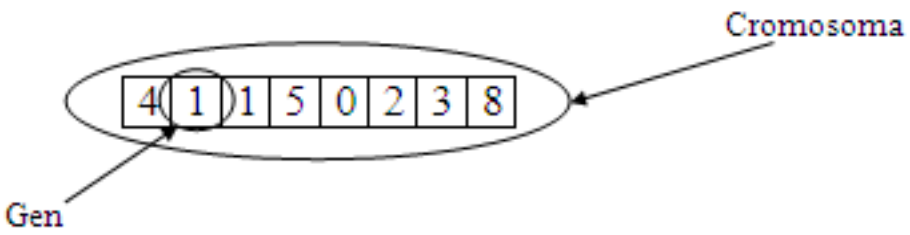

Ilustración 57 - Cromosoma

Este es un proceso aleatorio que proporciona una población de individuos que generalmente no permiten la obtención de resultados precisos. Ante esta situación, la población inicial se crea mediante la asignación de individuos aleatoriamente para completar los miembros necesarios en un equipo (número previamente indicado por el usuario del sistema). Gracias a la aleatoriedad, conseguimos que haya diversidad dentro de la población inicial evitando alcanzar una posible convergencia prematura y por tanto errónea.

Durante este proceso, la repetición de los individuos en el mismo equipo es controlado, ya que, no es posible que el mismo individuo aparezca dos veces en la población inicial. Una vez que la longitud de la población es adecuada y tenemos los individuos necesarios, cada uno de ellos formando parte de un equipo, es hora de pasar a la siguiente fase del algoritmo genético.

1.3.2 Evolución de los individuos y función fitness

En este segundo paso se evalúan los individuos de la población mediante la denominada función de fitness o función de evaluación, que nos indica cómo de bueno es el individuo con respecto a los demás, es decir, nos indica la solución al problema.

Este es el paso más importante de un algoritmo genético porque proporciona el valor que esperamos obtener, es decir, en nuestro caso nos dirá cómo de bueno es un equipo de trabajo. El resultado no va a ser siempre el que esperamos obtener, lo que nos ayudará a que el algoritmo genético evolucione favorablemente.

Tras el estudio de diferentes posibilidades en el desarrollo de la función de aprendizaje, finalmente se ha implementado un algoritmo genético mono-objetivo, cuya función fitness será obtenida a partir de la red de neuronas implementada en el apartado anterior. Se ha optado por esta solución ya que se ha considerado que es la mejor forma de gestionar los resultados, pudiendo analizar cada uno de los equipos y sus miembros de la mejor manera posible. Este procedimiento es repetido para todos los cromosomas de la población, de manera que obtendremos un valor de fitness por cada individuo.

Al aplicar la función fitness comparamos que no se estén metiendo individuos iguales, ya que durante las operaciones de cruce y mutación se podría dar la posibilidad de que se repitiera algún individuo, hecho que es imposible en la realidad ya que la misma persona no puede aparecer dos veces en el mismo equipo. Para solucionar este posible problema, cada individuo es identificado por un atributo identificador ID, de manera que así podremos comprobar que no se repita este valor más de una vez dentro de un mismo equipo.

Para analizar y clasificar los valores obtenidos, se ha establecido una clasificación ponderada para el equipo medida en porcentaje, proporcionando así una idea clara de lo bueno que es el equipo formado y su adecuación a los requisitos establecidos.

La función de fitness del algoritmo queda de la siguiente manera:

$$y = a_i^C = f\left(\sum_{j=1}^{n_{C-1}} w_{ji}^{C-1}\, a_j^{C-1} + u_i^C\right)$$

Donde

- y es la salida de la función
- a_i^C son las activaciones de cada capa
- w_{ji}^{C-1} es el peso de la neurona j de la capa i
- u_i^C es el umbral o bias de cada capa

1.3.3 Reproducción y selección

En este paso se seleccionan los individuos que en la fase anterior han obtenido un mejor fitness para decidir quiénes deben sobrevivir y que en el próximo paso sean cruzados (se reproduzcan).

El método implementado para la reproducción es el método torneo. Este método consiste en realizar la selección en base a comparaciones directas entre individuos de la población, seleccionando aquel que tenga mayor valor de fitness.

El tamaño de los torneos es un factor importante, ya que cuanto mayor sea la competencia, menor será la probabilidad de que los individuos menos adaptados sobrevivan. Este hecho parece positivo, sin embargo, podrían darse problemas de convergencia prematura como ya se ha mencionado anteriormente.

Existen dos versiones de este método, en nuestro caso, emplearemos la versión determinística.

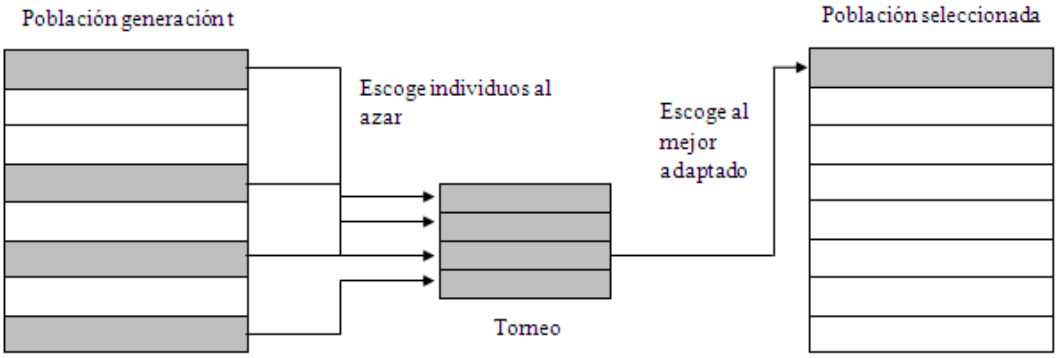

Ilustración 58 - Método Torneo

Se trata del método de selección por torneos más sencillo. El procedimiento de este método es el siguiente:

1. Seleccionamos al azar un número N de individuos
2. A partir de esos N individuos, seleccionamos a los dos individuos más aptos
3. Cruzo esos dos mejores individuos
4. Repetimos de nuevo el proceso

Es importante saber que el paso 4 se realiza con la población total, es decir, que se pueden, por casualidad, reproducir algunos varias veces, otros ninguna, ya que la selección de individuos que forman parte del torneo se hace de forma totalmente aleatoria.

1.3.4 Cruce

El objetivo de cruce es mezclar el código genético de los cromosomas para así obtener mejores individuos que en las generaciones anteriores. Para nuestro sistema, el cruce que realizamos es un *cruce uniforme*.

En este tipo de cruce, cada gen de la descendencia tiene las mismas probabilidades de pertenecer a cualquiera de los dos padres. El cruce uniforme consiste básicamente, en aplicar una máscara de cruce con valores binarios, es decir, si en una de las posiciones de la máscara hay un 1, el gen situado en esa posición en uno de los descendientes se copia del primer padre. Si por el contrario hay un 0 el gen situado en esa posición se copia del segundo padre. Para producir el segundo descendiente se intercambian los papeles de los padres, o dicho de otra manera, se intercambia la interpretación de los unos y los ceros de la máscara de cruce.

Como se puede observar en la siguiente figura, la descendencia contiene una mezcla de genes de cada uno de los padres. Por tanto, el número de puntos de cruce es fijo y será aproximadamente L/2 siendo L la longitud del cromosoma.

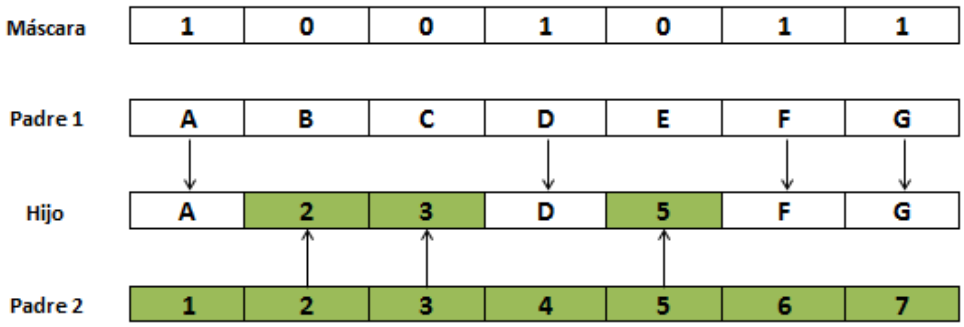

Ilustración 59 - Cruce AG

1.3.5 Mutación

La mutación es la operación que nos permite asegurar la diversidad genética ya que aleatoriamente cambiamos algunos genes del cromosoma permitiendo la aparición de individuos con nuevas características y evitando el estancamiento de la población.

En cada generación, hay una probabilidad de mutación de uno entre el tamaño de la población de que se produzca la mutación. Esta probabilidad debe ser muy baja de manera que la búsqueda de buenos individuos no sea ciega, es decir, si el número de mutaciones fuera muy elevado no se parecerán mucho los individuos de unas generaciones a otras y, por tanto, no se produciría una evolución progresiva.

En nuestro caso, si se produce la mutación, se selecciona un individuo al azar y seleccionamos sólo un gen de este individuo y se cambia. Es decir, se selecciona un equipo al azar y dentro de él se selecciona un miembro del equipo y se cambia por otro de manera aleatoria.

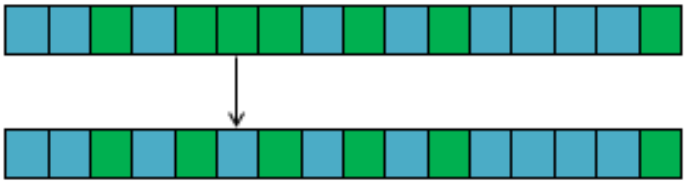

Ilustración 60 - Mutación AG

Todo nuestro algoritmo genético ha sido desarrollado en lenguaje Java, empleando para ello la herramienta Netbeans 7.2.1. (Anexo A).

2. Análisis de los resultados

Una vez diseñado todo el sistema, procedemos a analizar los resultados obtenidos por cada una de las partes.

2.1 Resultados para la red de neuronas

Una vez expuestos y representados las 3 arquitecturas desarrolladas, vamos a analizar los resultados obtenidos para cada una de ellas, con el fin de elegir cuál será la arquitectura idónea para nuestro sistema.

Los parámetros de configuración que hemos analizado para cada una de las tres topologías de red durante la fase de experimentación fueron:

- Tasa de aprendizaje
- Factor momento
- Valores iniciales de los pesos

Primero analizaremos el valor de inicialización de los pesos de la red neuronal. Estos pesos se generarán aleatoriamente dentro de un rango de valores. Este rango de valores será nuestro parámetro de estudio.

A continuación, procederemos a estudiar el factor momento y la tasa de aprendizaje variando sus posibles valores. El algoritmo backpropagation de nuestra red se estará ejecutando hasta que la red alcance un valor de error menos que el umbral de error prefijado o cuando se hayan completado el número de ciclos estipulados. Cuando la red llega a este umbral del error se considera que la red ha podido ser entrenada.

Una vez obtengamos los parámetros óptimos para nuestra red de neuronas, estos serán los que emplearemos para obtener la función fitness y aplicarla en nuestro algoritmo genético.

2.1.1 Evaluación previa

En cuanto a nuestro set de entrenamiento está compuesto por 40 equipos de desarrollo de software. Cada uno de estos equipos está formado por 5 miembros, y cada miembro tiene 8 características que lo definen.

Como ya se ha mencionado, estos atributos han sido normalizados dentro del intervalo [0,1] con el fin de simplificar el entrenamiento reduciendo los tiempos.

Este conjunto de datos se dividió a su vez en dos subconjuntos, uno para la fase de entrenamiento y otro para la fase de validación, siendo siempre menor el subconjunto de datos para la fase de validación.

	Tamaño de los conjuntos	Porcentajes
Conjunto total	40	100 %
Subconjunto entrenamiento	26	65 %
Subconjunto validación	14	35 %

Tabla 30 - Conjuntos entrenamiento

A priori puede parecer un conjunto de datos pequeño, sin embargo, dentro del ámbito para el que se desarrolla un proyecto, es un conjunto suficientemente grande ya que en la actualidad es difícil ver empresas con 40 equipos de desarrollo de software. Como se ha mencionado en el apartado 1.2.2 del capítulo 10, los datos han sido proporcionados por parte de Ericsson España.

Estos datos me han sido facilitados por formar parte de dicha empresa, sin embargo, dado que son datos de carácter corporativo y debido a la ley de protección de Datos, sólo se me ha proporcionado un subconjunto de toda la información posible.

En cuanto a cómo vamos a evaluar el resultado obtenido por la red, lo haremos comparando el MSE o error cuadrático medio. Se ha optado por el MSE porque este es inversamente proporcional al resultado obtenido por la red, es decir, un valor bajo del MSE supone un buen resultado de la red. Además, se ha elegido esta medida ya que es muy fácil de calcular, por lo que reducimos el coste.

2.1.2 Experimentación

El objetivo de este apartado es determinar cuáles serán los valores óptimos de los parámetros indicados para lograr entrenar nuestra red de neuronas. Durante todos los experimentos, los siguientes factores han permanecido constantes:

- Número de ciclos del algoritmo: 1000
- MSE máximo: 0.05
- Umbral de acierto: 0.25

2.1.2.1 Valores de inicialización

Para realizar las pruebas para estos factores, vamos a utilizar los siguientes valores para cada parámetro:

- Rangos = {[-0.01, 0.01]; [-0.1, 0.1]; [-1, 1]; [-5, 5]}

Los resultados que hemos obtenido son los siguientes:

- Para la arquitectura número 1:

Min Rango	Max Rango	Ciclos	Tiempo (seg)
-0.01	0.01	1000	68
-0.1	0.1	1000	80
-1	1	1000	67
-5	5	1000	102

Tabla 31 - Error/Tiempo según valores de los pesos 1

- Para la arquitectura número 2:

Min Rango	Max Rango	Ciclos	Tiempo (seg)
-0.01	0.01	1000	23
-0.1	0.1	1000	25
-1	1	1000	20
-5	5	1000	30

Tabla 32 - Error/Tiempo según valores de los pesos 2

- Para la arquitectura número 3:

Min Rango	Max Rango	Ciclos	Tiempo (seg)
-0.01	0.01	1000	35
-0.1	0.1	1000	40
-1	1	1000	35
-5	5	1000	45

Tabla 33 - Error/Tiempo según valores de los pesos 3

Una vez realizados todos los experimentos procedemos a analizar los resultados obtenidos. En la siguiente tabla mostraremos los mejores resultados para cada topología de la red y los compararemos entre ellos:

Topología de red	Min Rango	Max Rango	Tiempo (seg)
Arquitectura 1	-1	1	67
Arquitectura 2	-1	1	20
Arquitectura 3	-1	1	35

Tabla 34 – Resultado Error/Tiempo según valores de los pesos

A partir de la tabla anterior, y en función a los resultados obtenidos para las tres arquitecturas, podemos concluir lo siguiente:

- Si los valores de inicialización de los pesos son muy altos, el nivel de entrenamiento de la red es más bajo y además el tiempo que tarda en aprender es mayor. (*Tabla 31*, *Tabla 32* y *Tabla 33*)
- Si los valores de inicialización de los pesos son muy bajos el tiempo de entrenamiento es menor. (*Tabla 31*, *Tabla 32* y *Tabla 33*)

- Los valores de inicialización que obtuvieron mejor resultado fueron para el rango [-1,1]. (*Tabla 34*)

2.1.2.2 Tasa de aprendizaje y factor momento

Para realizar las pruebas para estos factores, vamos a utilizar los siguientes valores para cada parámetro, como se indicó en el apartado 1.2.4 del capítulo 10:

- Tasa de aprendizaje (α) = {0.05, 0.1, 0.25, 0.5}
- Factor momento (β) = {0, 0.25, 0.5, 0.9}

Los resultados que hemos obtenido son los siguientes:

- Para la arquitectura número 1:

Aprendizaje	Momento	Ciclos	Tiempo (seg)	% Error entrenamiento
0.05	0	1000	98	0.051576
	0.25	1000	106	0.051562
	0.5	1000	100	0.048243
	0.9	1000	97	0.006729

Tabla 35 - Error/Tiempo con T.aprendizaje=0.05 1

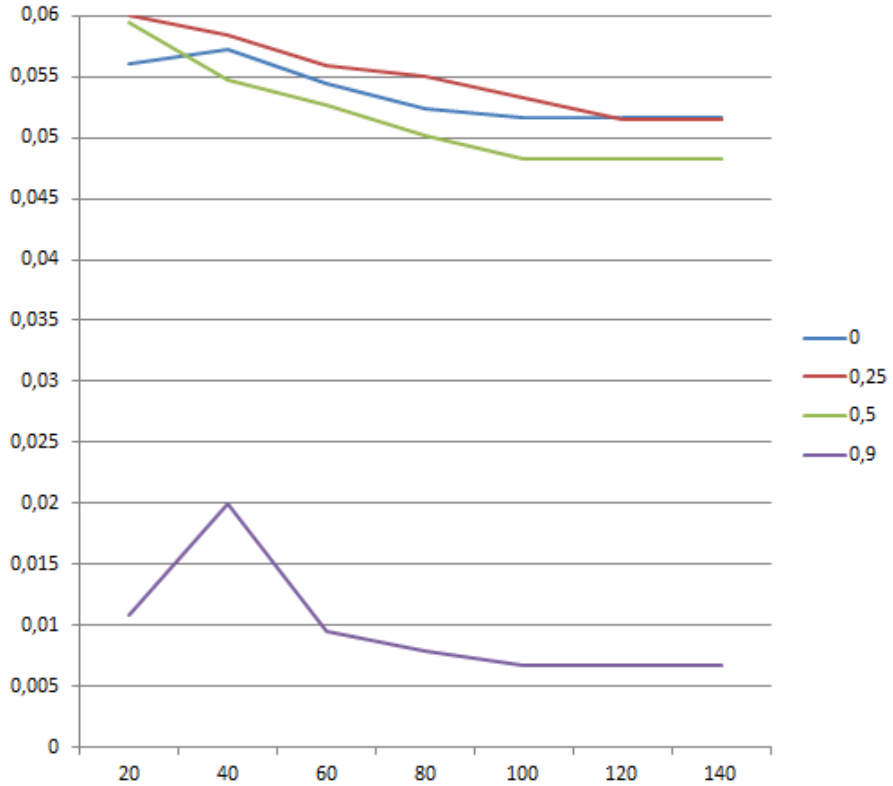

Ilustración 61 - Error/Tiempo con T.aprendizaje=0.05 1

Aprendizaje	Momento	Ciclos	Tiempo (seg)	% Error entrenamiento
0.1	0	1000	75	0.007328
	0.25	1000	89	0.002189
	0.5	1000	82	0.001143
	0.9	1000	79	0.000404

Tabla 36 - Error/Tiempo con T.aprendizaje=0.1 1

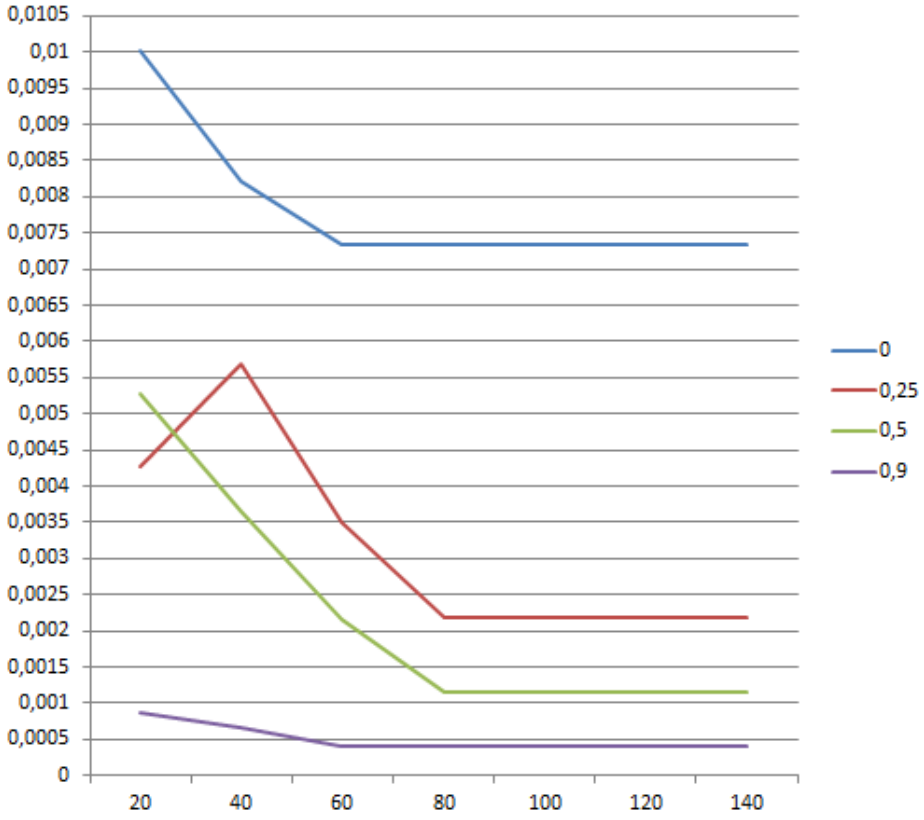

Ilustración 62 - Error/Tiempo con T.aprendizaje=0.1 1

Aprendizaje	Momento	Ciclos	Tiempo (seg)	% Error entrenamiento
0.25	0	1000	70	0.000982
	0.25	1000	67	0.000767
	0.5	1000	69	0.000603
	0.9	1000	68	0.001055

Tabla 37 - Error/Tiempo con T.aprendizaje=0.25 1

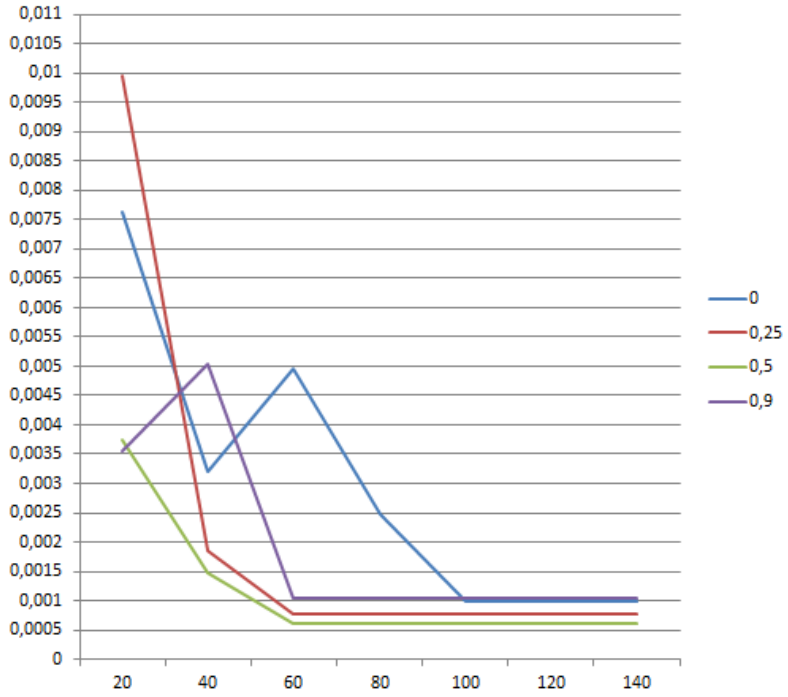

Ilustración 63 - Error/Tiempo con T.aprendizaje=0.25 1

Aprendizaje	Momento	Ciclos	Tiempo (seg)	% Error entrenamiento
0.5	0	1000	118	0.000621
	0.25	1000	115	0.000546
	0.5	1000	122	0.000465
	0.9	1000	121	0.000782

Tabla 38 - Error/Tiempo con T.aprendizaje=0.5 1

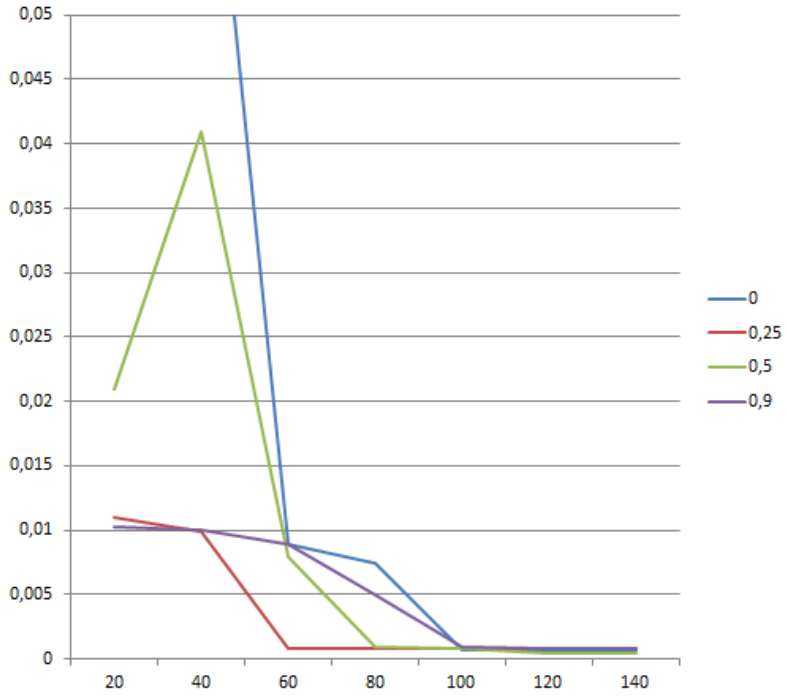

Ilustración 64 - Error/Tiempo con T.aprendizaje=0.5 1

- Para la arquitectura número 2:

Aprendizaje	Momento	Ciclos	Tiempo (seg)	% Error entrenamiento
0.05	0	1000	95	0.051498
	0.25	1000	100	0.051690
	0.5	1000	97	0.049253
	0.9	1000	92	0.000729

Tabla 39 - Error/Tiempo con T.aprendizaje=0.05 2

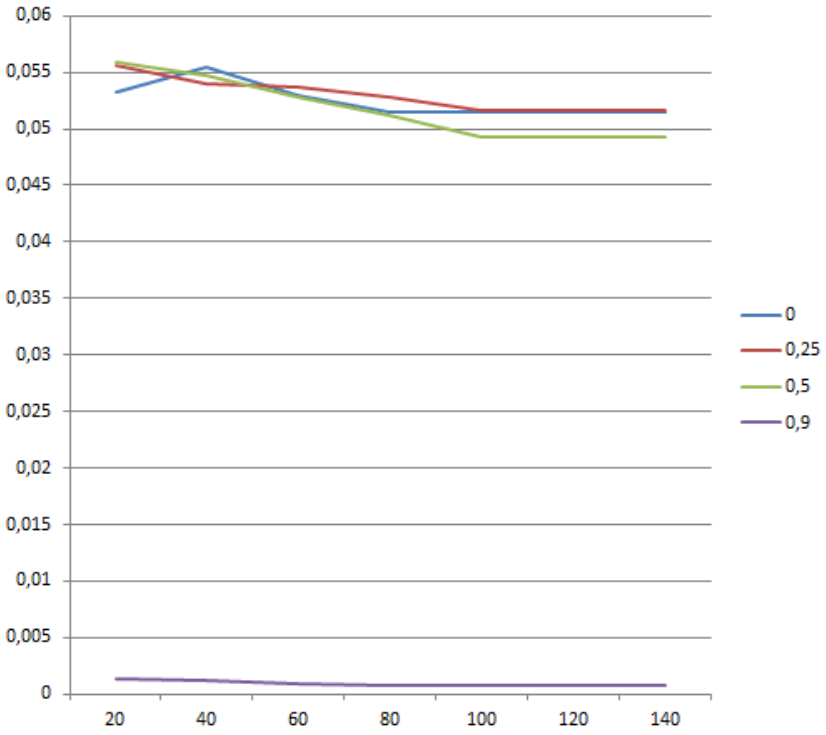

Ilustración 65 - Error/Tiempo con T.aprendizaje=0.05 2

Aprendizaje	Momento	Ciclos	Tiempo (seg)	% Error entrenamiento
0.1	0	1000	76	0.051498
	0.25	1000	79	0.006916
	0.5	1000	84	0.001925
	0.9	1000	78	0.000507

Tabla 40 - Error/Tiempo con T.aprendizaje=0.1 2

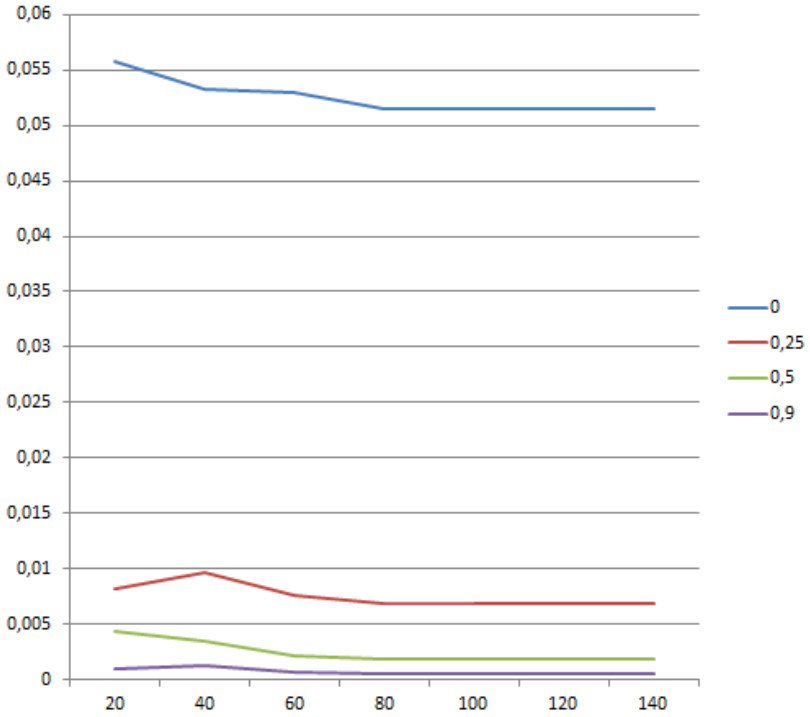

Ilustración 66 - Error/Tiempo con T.aprendizaje=0.1 2

Aprendizaje	Momento	Ciclos	Tiempo (seg)	% Error entrenamiento
0.25	0	1000	72	0.001475
	0.25	1000	70	0.001021
	0.5	1000	68	0.000767
	0.9	1000	65	0.000657

Tabla 41 - Error/Tiempo con T.aprendizaje=0.25 2

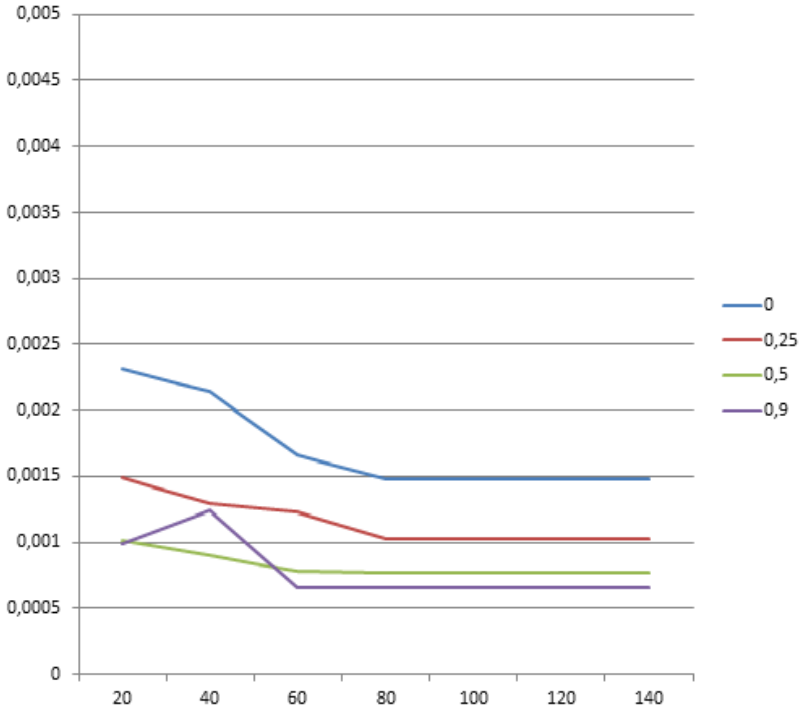

Ilustración 67 - Error/Tiempo con T.aprendizaje=0.25 2

Aprendizaje	Momento	Ciclos	Tiempo (seg)	% Error entrenamiento
0.5	0	1000	119	0.000800
	0.25	1000	116	0.000688
	0.5	1000	120	0.000571
	0.9	1000	119	0.001017

Tabla 42 - Error/Tiempo con T.aprendizaje=0.5 2

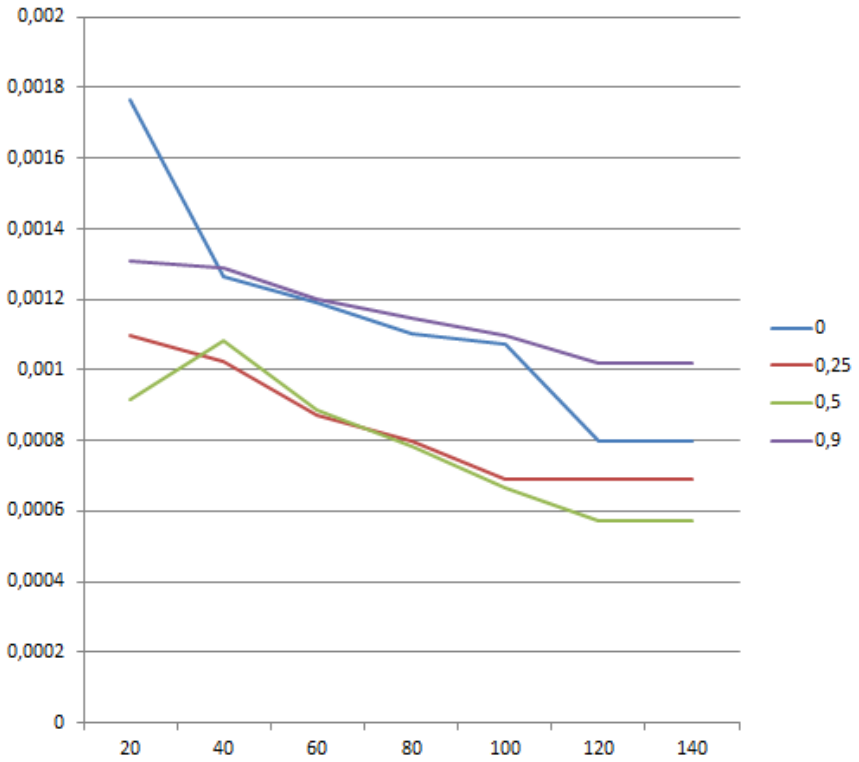

Ilustración 68 - Error/Tiempo con T.aprendizaje=0.5 2

- Para la arquitectura número 3:

Aprendizaje	Momento	Ciclos	Tiempo (seg)	% Error entrenamiento
0.05	0	1000	97	0.051744
	0.25	1000	99	0.052194
	0.5	1000	94	0.049382
	0.9	1000	100	0.000768

Tabla 43 - Error/Tiempo con T.aprendizaje=0.05 3

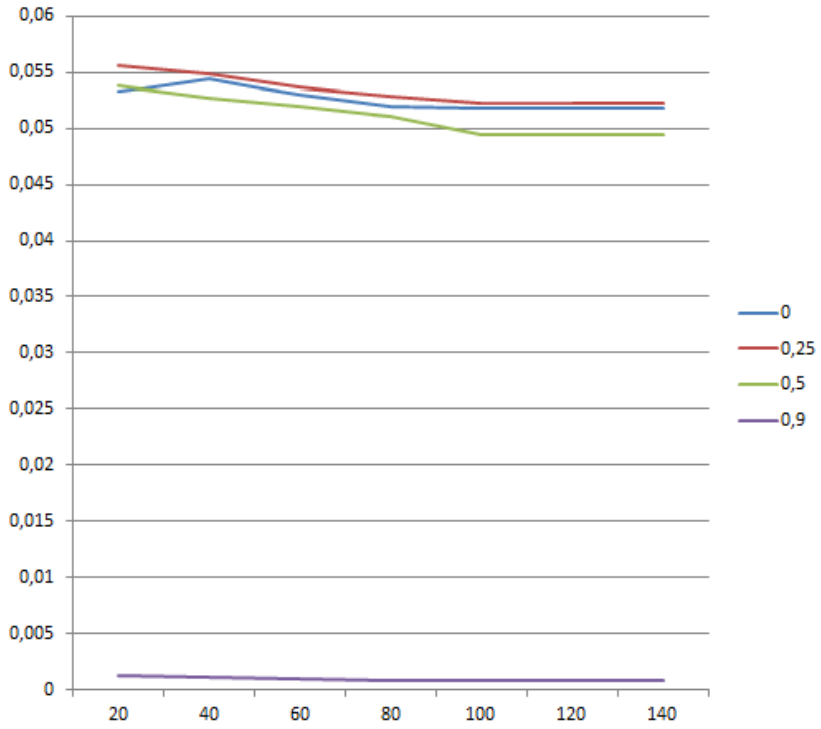

Ilustración 69 - Error/Tiempo con T.aprendizaje=0.05 3

Aprendizaje	Momento	Ciclos	Tiempo (seg)	% Error entrenamiento
0.1	0	1000	75	0.051889
	0.25	1000	80	0.005548
	0.5	1000	82	0.001382
	0.9	1000	73	0.000508

Tabla 44 - Error/Tiempo con T.aprendizaje=0.1 3

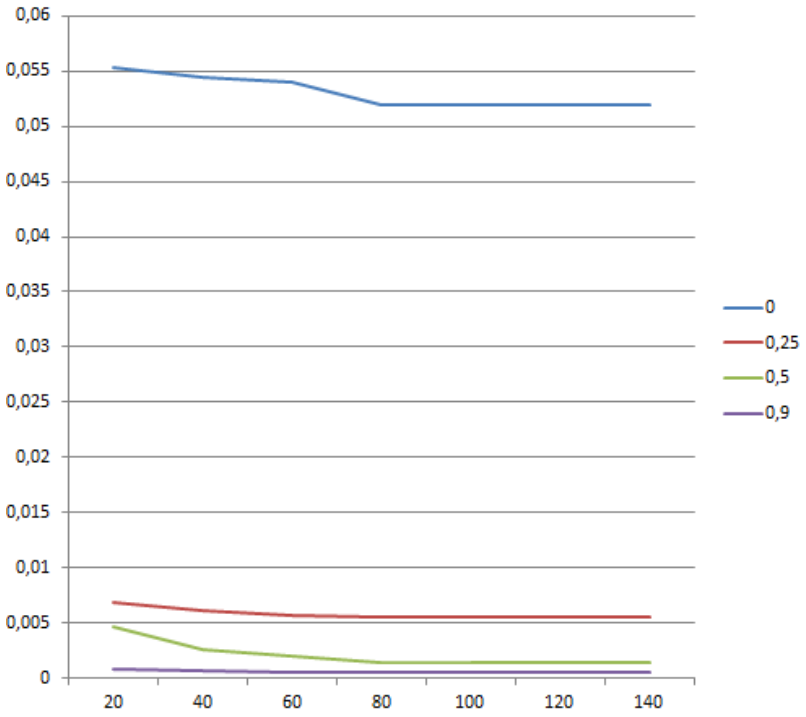

Ilustración 70 - Error/Tiempo con T.aprendizaje=0.1 3

Aprendizaje	Momento	Ciclos	Tiempo (seg)	% Error entrenamiento
0.25	0	1000	71	0.001114
	0.25	1000	68	0.000851
	0.5	1000	75	0.000792
	0.9	1000	83	0.000336

Tabla 45 - Error/Tiempo con T.aprendizaje=0.25 3

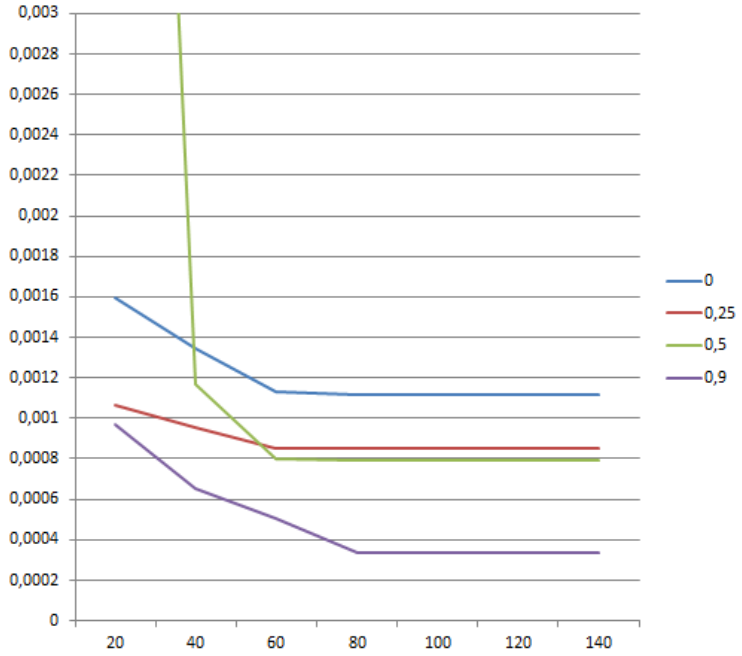

Ilustración 71 - Error/Tiempo con T.aprendizaje=0.25 3

Aprendizaje	Momento	Ciclos	Tiempo (seg)	% Error entrenamiento
0.5	0	1000	118	0.000898
	0.25	1000	122	0.000710
	0.5	1000	119	0.000584
	0.9	1000	122	0.000453

Tabla 46 - Error/Tiempo con T.aprendizaje=0.5 3

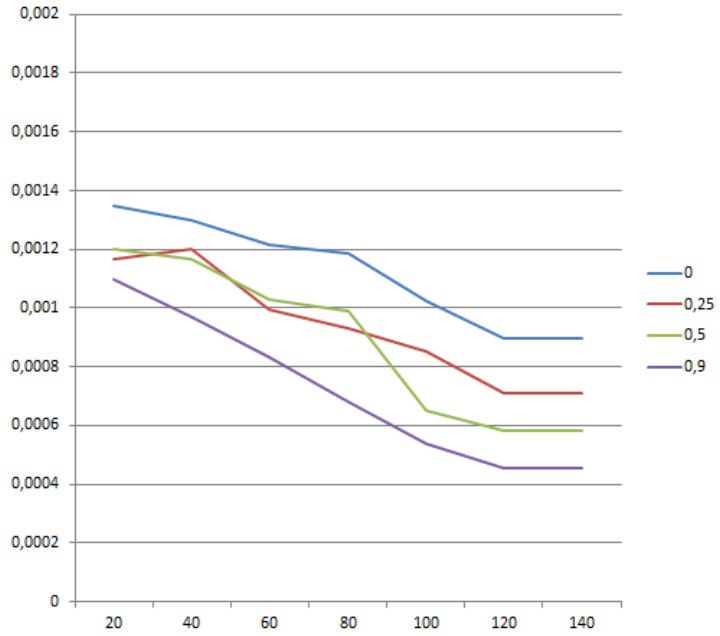

Ilustración 72 - Error/Tiempo con T.aprendizaje=0.5 3

Una vez realizados todos los experimentos procedemos a analizar los resultados obtenidos. En la siguiente tabla mostraremos los mejores resultados para cada topología de la red y los compararemos entre ellos:

Topología de red	Aprendizaje	Momento	Tiempo (seg)	% Error entrenamiento
Arquitectura 1	0.1	0.9	79	0.000404
Arquitectura 2	0.1	0.9	78	0.000507
Arquitectura 3	0.25	0.9	83	0.000336

Tabla 47 – Resultado Error/Tiempo con variación de tasas

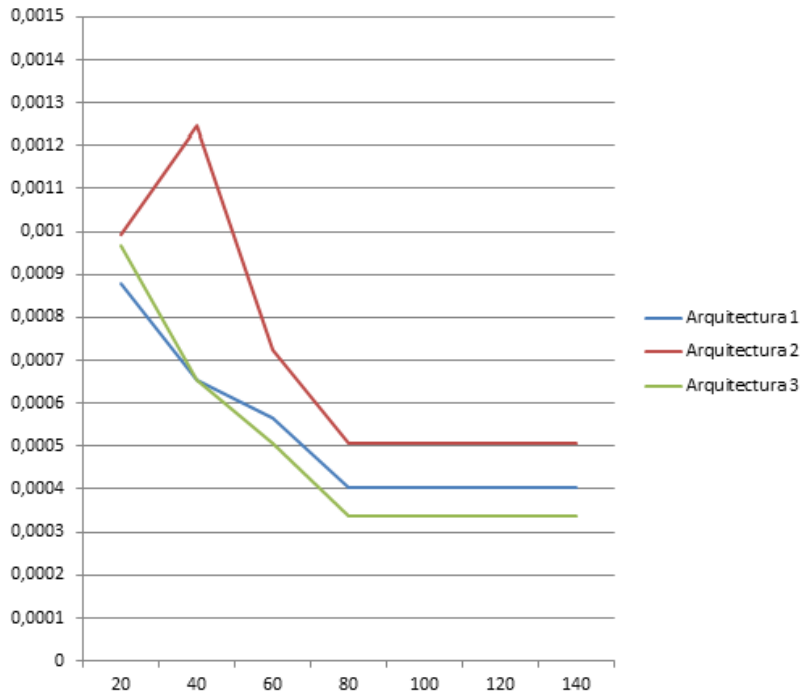

Ilustración 73 - – Resultado Error/Tiempo con variación de tasas

A partir de la tabla anterior podemos afirmar que la mejor topología es la tercera, y en función a los resultados de ésta, podemos concluir lo siguiente:

- Si el valor de la tasa de aprendizaje es bajo el proceso hasta alcanzar el punto de convergencia es lento. Sin embargo, con estos valores bajos se alcanzan mejores valores de error mínimo. Cuando la tasa de aprendizaje es baja, la probabilidad de alcanzar la convergencia es alta y viceversa.
- Cuando el valor del factor momento es próximo a 1, el aprendizaje de la red mejora considerablemente obteniendo también los mejores valores de error. Como podemos observar en la *Tabla 47*, los mejores resultados de error se obtienen para un factor momento de 0.9.
- El mejor porcentaje de error durante la fase de entrenamiento fue recogido para una tasa de aprendizaje de 0.25 y un factor momento de 0.9 (ver *Tabla 45*) lo que se corresponde con la teoría de Rumelhart mencionada en el apartado 1.2.4 del capítulo 10.
- En cuanto al tiempo cabe destacar que dado el tamaño de nuestro conjunto de datos, en cualquiera de las tres topologías de red se obtienen resultados de tiempo buenos, y a pesar de que en el caso de la tercera topología de red, el tiempo de entrenamiento es algo mayor que el resto sigue siendo un valor suficientemente bueno.

Por lo tanto, siguiendo los resultados obtenidos en la *Tabla 34* y en la *Tabla 47*, la topología elegida para nuestra red de neuronas es la **arquitectura 3** formada por 25 neuronas en la primera capa oculta y 20 en la segunda capa oculta, ya que aunque la tasa de aprendizaje es algo mayor que en los otros dos casos, sigue siendo un valor pequeño que está dentro del rango de los valores óptimos de la tasa de aprendizaje. Sera esta arquitectura la que nos proporcione la función fitness de nuestro algoritmo genético.

Capítulo 11 | Diseño interfaz de usuario

Una vez que hemos desarrollado el funcionamiento del sistema internamente, en este apartado vamos a mostrar la apariencia del sistema de cara al usuario.

A continuación, vamos a exponer las interfaces de usuario que van a formar parte del sistema, y mediante las cuales el usuario será capaz de interactuar con él.

Los diseños de las interfaces que se exponen a continuación, carecen de funcionalidad, y tienen como único objetivo mostrar visualmente la manera en la que el usuario interactuará con las interfaces. Todas las interfaces han sido diseñadas para cumplir con los requisitos de usabilidad y facilidad de uso.

La primera interfaz que ofrecerá el sistema al usuario es la página de inicio, en la cual el usuario deberá introducir su nombre de usuario y contraseña para poder acceder a la funcionalidad del sistema.

Ilustración 74 - Interfaz Inicio

Una vez que el usuario es autenticado por el sistema, accederá a la segunda interfaz. En esta interfaz, se muestra el sistema para la empresa a la que pertenece el usuario, es decir, toda la funcionalidad que desee realizar este usuario en el sistema podrá ser realizada con la información de la base de datos que esté asociada a dicha empresa.

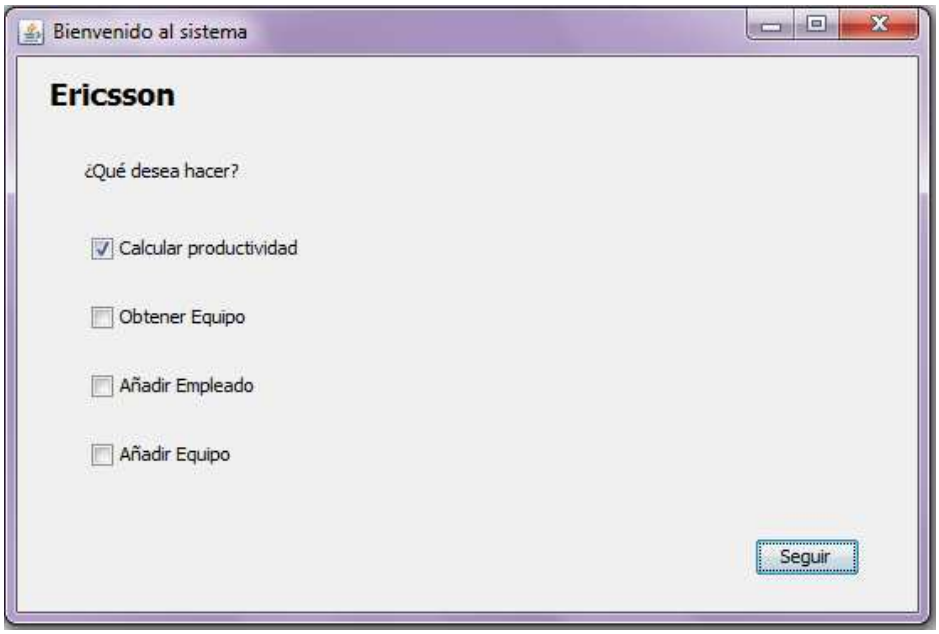

Ilustración 75 - Interfaz Acciones

En la interfaz anterior, se le proporcionan al usuario todas las distintas acciones que puede realizar a través del sistema, cada una de las cuales tendrá asociada una interfaz. Para seleccionar la acción deseada, bastará con que el usuario selecciona la casilla asociada a cada acción.

- **Calcular Productividad**

Si el usuario selecciona la primera acción: “Calcular productividad”, y hace clic en el botón “Seguir”, accederá a la siguiente interfaz:

Ilustración 76 - Interfaz Productividad

En esta interfaz, el usuario debe seleccionar el equipo de trabajo para el que desea calcular la productividad, así como el proyecto para el cual quiere calcularla. Tanto los equipos de desarrollo que forman parte de la lista como los proyectos que existen en la empresa serán mostrados como aparece en la interfaz a modo de lista.

Una vez seleccionados ambos campos, el usuario deberá hacer clic en el botón "Productividad" para que el sistema proceda a calcularla.

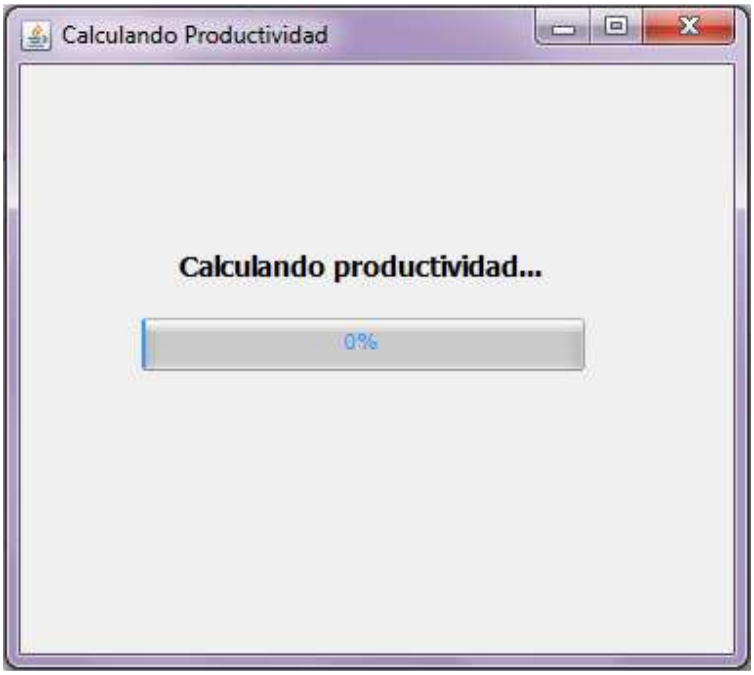

Ilustración 77 - Interfaz Calculando productividad

Ilustración 78 - Interfaz Resultado productividad

Las Figuras 77 y 78 muestran por un lado, el proceso del sistema mientras calcula la productividad del equipo seleccionado (*Ilustración 77* - Interfaz Calculando productividad), y por otro lado, el resultado final obtenido por el sistema sobre la productividad obtenida, mostrando tanto el equipo seleccionado como el proyecto en cuestión (*Ilustración* 78 - Interfaz Resultado productividad).

- **Obtener Equipo**

En la *Ilustración 75* - Interfaz Acciones, podemos seleccionar una segunda opción que es "Obtener equipo". Esta opción da la posibilidad al usuario calcular la mejor combinación de empleados para un proyecto de manera que esa formación genere una mayor productividad.

Entonces, el sistema de manera automática y a partir de todos los empleados de la empresa almacenados en el sistema calculará la mejor formación de un equipo para un determinado proyecto.

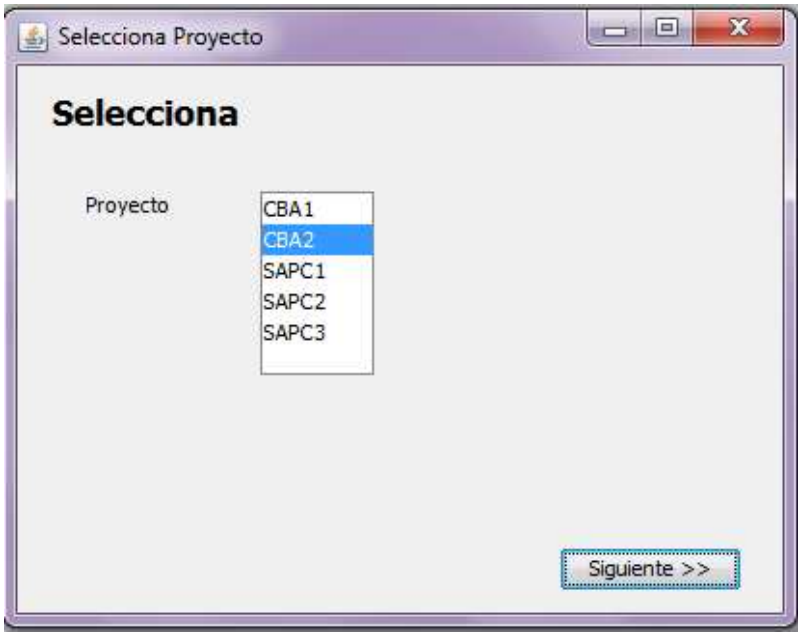

Ilustración 79 - Interfaz Selección proyecto

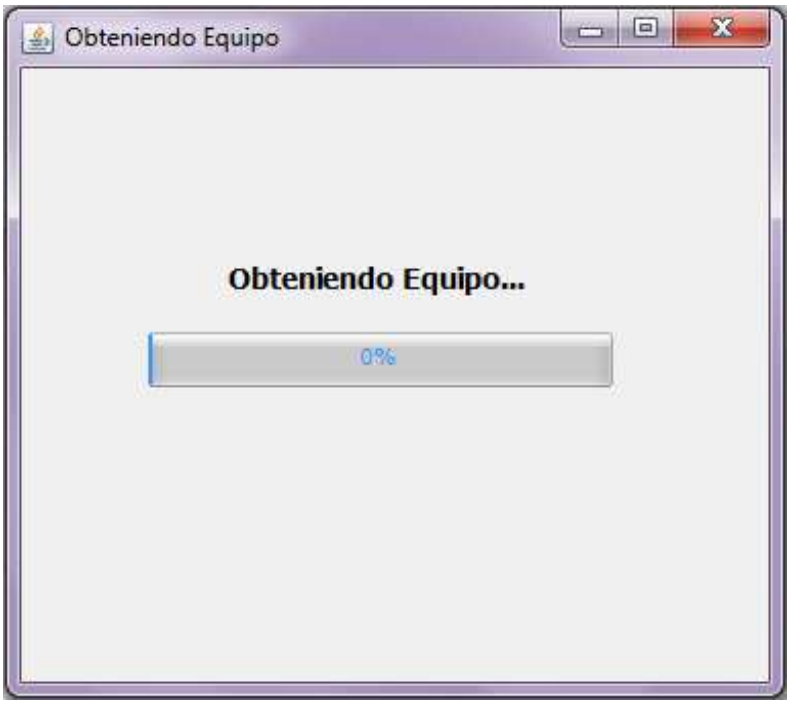

Ilustración 80 - Interfaz Obtener Equipo

Las figuras 79 y 80 muestran el proceso en el que el usuario selecciona el proyecto para el cual va a buscar un equipo de desarrollo (*Ilustración 79* - Interfaz Selección proyecto), así como el proceso de búsqueda del sistema de dicho equipo (*Ilustración 80* - Interfaz Obtener Equipo).

Tras el proceso de ejecución del sistema en el que busca el mejor equipo para un determinado proyecto, el sistema mostrará dicha formación.

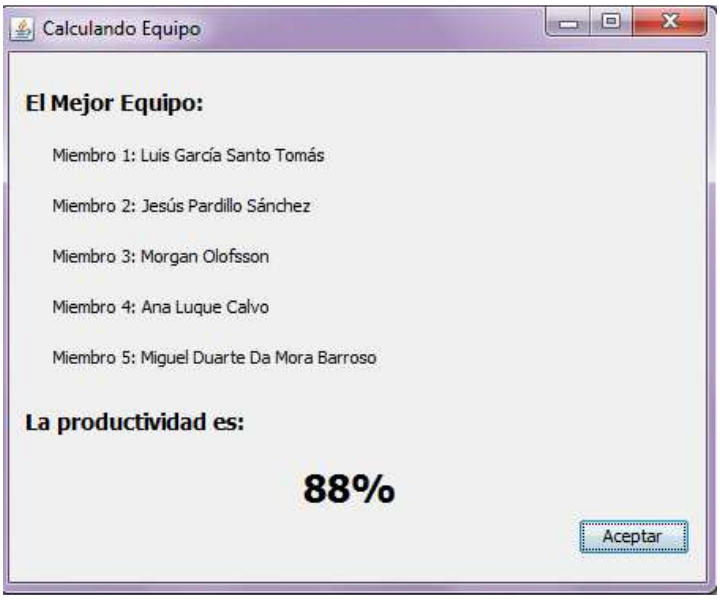

Ilustración 81 - Interfaz Resultado equipo

- **Añadir Empleado**

En la *Ilustración 75* - Interfaz Acciones, podemos seleccionar una tercera opción que es la de "Añadir empleado". Esta opción será empleada por el usuario, en situaciones como una nueva incorporación a la plantilla de la empresa, de manera que para posteriores búsquedas de la mejor formación de un equipo para un determinado proyecto, esta nueva incorporación pueda ser tenida en cuenta por el sistema.

La interfaz que mostrará el sistema al usuario en caso de seleccionar esta opción es la siguiente:

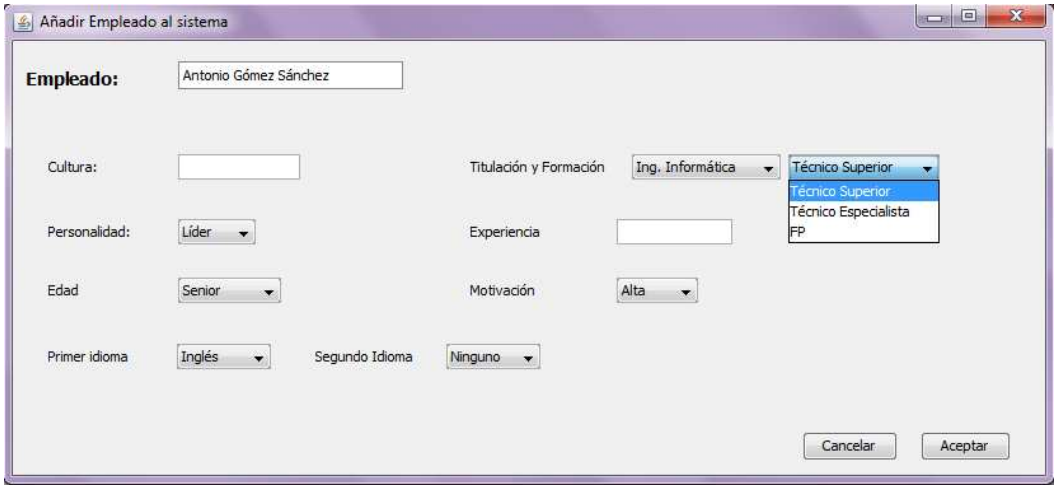

Ilustración 82 - Interfaz Añadir empleado

- **Añadir Equipo**

Para que el usuario pueda calcular la productividad de un equipo, este deberá formar parte del sistema, es decir, si se crea un nuevo equipo de desarrollo en la empresa, este deberá ser introducido en el sistema previamente, para que así pueda aparecer en la lista de equipos de la *Ilustración 76* - Interfaz Productividad

Para añadir un equipo, el usuario en la interfaz mostrada en la *Ilustración 75* - Interfaz Acciones, deberá seleccionar la acción "Añadir Equipo", y el sistema le llevará a la siguiente interfaz:

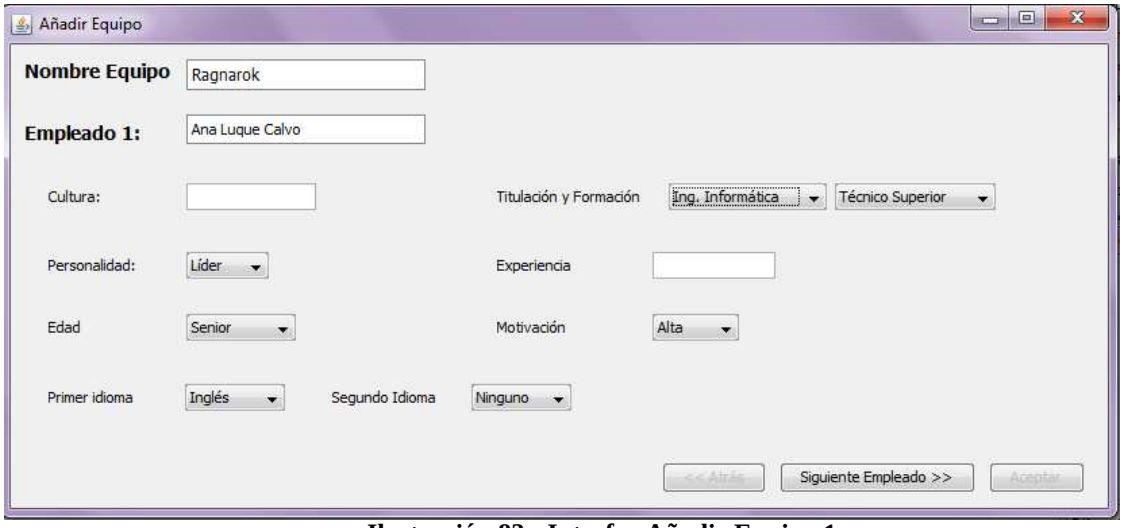

Ilustración 83 - Interfaz Añadir Equipo 1

En esta interfaz podemos observar que la primera acción que debe realizar el usuario es asignarle un nombre al equipo, y a continuación proceder a añadir los miembros de dicho equipo, introduciendo cada uno de los atributos que utilizará nuestro sistema para el cálculo de la productividad.

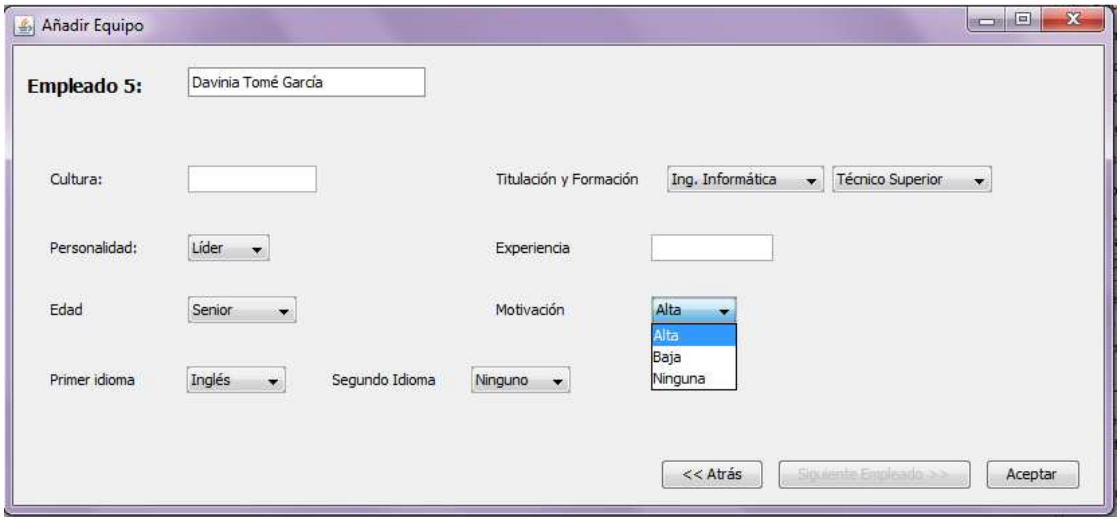

Ilustración 84 - - Interfaz Añadir Equipo 2

El proceso de añadir un equipo finaliza tras introducir al quinto miembro del equipo, y pulsando el botón "Aceptar". De esta manera podremos proceder posteriormente si se deseara a calcular la productividad de este equipo.

Capítulo 12 | Trabajos futuros

Una vez finalizada la fase de implementación y simulación, en esta sección del documento, vamos a tratar de analizar las posibles vías de desarrollo que han quedado abiertas para continuar nuestro proyecto.

Las futuras líneas de investigación son las siguientes:

- Las mejoras de nuestro sistema, tras los resultados obtenidos durante fase de simulación y experimentación, se deben centrar en el perfeccionamiento de la función fitness de nuestro algoritmo para que los resultados obtenidos se acerquen o superen los obtenidos por nuestro sistema. De esta manera, se podría plantear distintas arquitecturas de redes de neuronas con el fin de analizar y estudiar exhaustivamente la mejor solución al sistema.

- Se podrían desarrollar la funcionalidad necesaria en el sistema que fuera capaz de generar automáticamente las gráficas de los resultados obtenidos por el sistema. Generando un histórico de todas las pruebas realizadas y la comparación de dichos resultados. Una opción es la posibilidad de exportar los resultados obtenidos a Excel.

- Ampliar el ámbito del sistema. Este sistema se centra en las características que se han considerado más relevantes para el funcionamiento y productividad de un equipo. Pero la introducción de nuevas variables daría un resultado mucho más óptimo y real.

PART VI: REFERENCIAS

Capítulo 13 | Referencias

En este capítulo vamos a recoger todas las referencias que nos han ayudado durante el desarrollo del proyecto. El formato en el que están indicadas sigue la normativa IEEE: 118

[1] Lazear, Edward P. Personnel Economics for Managers, New York, NY: John M. Wiley & Sons Inc., 1998a.

[2] Lazear, Edward P. "Globalization and the Market for Teammates," National Bureau of Economic Research working paper 6579, 1998b.

[3] http://www.cprime.com/about/scrum_faq.html

[4] http://sorprendemos.com/consultoresdocumentales/?p=4659

[5] Hofstede, Geert. Cultures & organizations: Software of the mind: Intercultural cooperation and its importance for survival. New York: McGraw-Hill, 1991.

[6] Knight D, Pearce CL, Smith KG, Olian JD, Sims HP, Smith KA, Flood P. «Top management team diversity, group process, and strategic consensus» Strategic Management Journal, vol. 20:445-6, 1999.

[7] Pelled, L.H., Eisenhardt, K.M. and Xin, K.R., "Exploring the Black Box: An Analysis of Work Group Diversity, Conflict, and Performance", Administrative Science Quarterly, Vol. 44(1), 1999.

[8] Vroom, V. H. Work and motivation. New York: Wiley, 1964.

[9] Reagans, R., Argote, L., & Brooks, D. Individual experience and experience working together: Predicting learning rates from knowing who knows what and knowing how to work together. Management Science, 51: 869-881, 2005.

[10] Huckman, R. S., Staats, B. R., & Upton, D. M. Team familiarity, role experience, and performance: Evidence from Indian software services. Management Science, 55: 85-100, 2009.

[11] Espinosa, J. A., Slaughter, S. A., Kraut, R. E. & Herbsleb, J. D. Familiarity, complexity, and team performance in geographically distributed software development. Organization Science, 18: 613-630, 2007.

[12] R.A.E. Diccionario de la lengua española. Vigésima segunda edición 2001. Ed. Espasa.

[13] Redes de Neuronas Artificiales: Un enfoque práctico. Pedro Isasi e Inés M. Galván. Pearson. Prentice Hall. Madrid 2004.

[14] Hilera J. y Martínez V. “Redes Neuronales Artificiales. Fundamentos, modelos y aplicaciones”, p: 9-10, 1995.

[15] Castillo E. y Cobo A. “Introducción a la redes funcionales con aplicaciones: un nuevo paradigma neuronal”, Paraninfo, p: 9-15, 1999.

[16] Moreno A. y Alonso J. “Aprendizaje automático” Edicions de la Universitat Politécnica de Catalunya, p: 160-162, 1994.

[17] Rojas R., “Neural Network”,Neural Networks, Springer-Verlag, Berlin, p: 151-184, 1996.

[18] “Aprendizaje Automático: conceptos básicos y avanzados: aspectos prácticos utilizando el software Weka”, Basilio Sierra Araujo, ISBN: 848322318X. Pearson Prentice Hall, 2006.

[19] K. A. De Jong. ``An analysis of the behavior of a class of genetic adaptive system". Tesis Doctoral. Universidad de Michigan, 1975.

[20] http://pymecrunch.com/scrum-metodologia-agil-para-tus-proyectos

[21] Roger S. Pressman. Software Engineering: A Practioner's Approach (Sixth Edition, International Edition). McGraw-Hill, 2005.

[22] Masters, T. Practical neural network recipes in C++. San Diego, CA, USA: Academic Press Professional, Inc, 1993

[23] SPSS Inc., 1997.

[24] Martín del Brío, B. y Sanz Molina, A. Redes Neuronales y sistemas borrosos, pp. 127-132, Ra-Ma, Madrid, 1997.

[25] Sarle, W.S.. Neural networks and statistical models. En SAS Institute (Ed.), Proceedings of the 19th Annual SAS Users Group International Conference (pp.1538-1550). Cary, NC: SAS Institute, 1994.

[26] www.ericsson.com/es

[27] Rumelhart, D. E., Hinton, G. E. & Williams, R. J. in Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. **1**: Foundations (eds Rumelhart, D. E. & McClelland, J. L.) 318−362 (MIT, Cambridge) 1986.

[28] http://eva.evannai.inf.uc3m.es/

[29] http://es.wikipedia.org/wiki/Henry_Ford

[30] http://www.giaa.inf.uc3m.es/docencia/ITIG/ReferenciaIEEE.pdf

Anexo A

En este Anexo vamos a describir brevemente todas las herramientas utilizadas para el desarrollo del sistema.

1. JustNNS

JustNNS se trata de una aplicación que provee un entorno sencillo y fácil de usar para el desarrollo e investigación de las redes neuronales.

Esta herramienta permite:

- Importar texto, csv, hojas de cálculo, imágenes y ficheros binarios en la tabla de datos.
- Usar distintos editores y funciones en la tabla de datos.
- Construir redes de neuronas a partir de la tabla de datos.
- Entrenar, validar y consultar la red de neuronas.

2. Java

Java es un lenguaje de programación multiplataforma basado en el paradigma de orientación a objetos. Este lenguaje creado, por Sun Microsystems en la década de los 90 ha sido aceptado con gran éxito, tanto es así que actualmente más de 4.500 millones de dispositivos utilizan la tecnología.

Java ha sido probado, mejorado, ampliado y probado por una comunidad especializada de más de 6,5 millones de desarrolladores, la mayor y más activa del mundo. Gracias a su versatilidad, eficiencia y portabilidad, Java se ha convertido en un recurso inestimable ya que permite a los desarrolladores:

- Desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma
- Crear programas para que funcionen en un navegador web y en servicios web
- Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML, etc.
- Combinar aplicaciones o servicios que usan el lenguaje Java para crear servicios o aplicaciones totalmente personalizados
- Desarrollar potentes y eficientes aplicaciones para teléfonos móviles, procesadores remotos, productos de consumo de bajo coste y prácticamente cualquier tipo de dispositivo digital

3. Netbeans 7.2.1

Se trata de un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java.

NetBeans es un proyecto de código abierto y gratuito de gran éxito fundado por Sun MicroSystems en junio de 2000 y continúa siendo el patrocinador principal de los proyectos.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

La primera versión de NetBeans fue la 3.5 que mejoró enormemente en desempeño, y con la llegada de NetBeans 3.6, se reimplementó el sistema de ventanas y la hoja de propiedades, y se limpió enormemente la interfaz. NetBeans 4.0 fue un gran cambio en cuanto a la forma de funcionar del IDE, con nuevos sistemas de proyectos, con el cambio no solo de la experiencia de usuario, sino del reemplazo de muchas piezas de la infraestructura que había tenido NetBeans anteriormente. NetBeans IDE 5.0 introdujo un soporte mucho mejor para el desarrollo de nuevos módulos, el nuevo constructor intuitivo de interfaces Matisse, un nuevo y rediseñado soporte de CVS, soporte a Sun ApplicationServer 8.2, Weblogic9 y JBoss 4.0. Con Netbeans 6.01, 6.8 y ahora todo perfecto en su versión mejorada 7.0 Se dio soporte a frameworks comerciales como son Struts.

4. Visual paradigm para UML

Visual Paradigm para UML es una herramienta CASE que soporta el modelado mediante UML y proporciona asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software.

Las ventajas que proporciona Visual Paradigm para UML son:

- Dibujo. Facilita el modelado de UML, ya que proporciona herramientas específicas para ello. Esto también permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado por la herramienta.
- Corrección sintáctica. Controla que el modelado con UML sea correcto.
- Coherencia entre diagramas. Al disponer de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, evitando duplicidades.
- Integración con otras aplicaciones. Permite integrase con otras aplicaciones, como herramientas ofimáticas, lo cual aumenta la productividad.
- Trabajo multiusuario. Permite el trabajo en grupo, proporcionando herramientas de compartición de trabajo.
- Reutilización. Facilita la reutilización, ya que disponemos de una herramienta centralizada donde se encuentran los modelos utilizados para otros proyectos.
- Generación de código. Permite generar código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación del software.
- Generación de informes. Permite generar diversos informes a partir de la información introducida en la herramienta.